Ontology Integration: Approaches and Challenging Issues

Inès Osman, Sadok Ben Yahia, Gayo Diallo

Highlights

Ontology Integration: Approaches and Challenging Issues

Inès Osman, Sadok Ben Yahia, Gayo Diallo

- Eliciting key definitions and notions related to the ontology integration area.
- Reporting and analyzing integration principles, consequences and repair techniques.
- Briefly reviewing the most relevant research works on the ontology integration area.
- Performing a holistic ontology integration to highlight open issues from the survey.
- Improving the ontology matching will considerably improve the ontology integration.

Ontology Integration: Approaches and Challenging Issues

Inès Osman[a,*], Sadok Ben Yahia[b], Gayo Diallo[c]

[a] *LIPAH - LR11ES14, Faculty of Sciences of Tunis, University of Tunis El Manar, 2092 Tunis, Tunisia*
[b] *Department of Software Science, Tallinn University of Technology, Estonia*
[c] *INRIA SISTM, Team ERIAS - INSERM Bordeaux Population Health Research Center, University of Bordeaux, F-33000 Bordeaux, France*

Abstract

In recent years, the decentralized development of ontologies has led to the generation of multiple ontologies of overlapping knowledge. This heterogeneity problem can be tackled by integrating existing ontologies to build a single coherent one. Ontology integration has been investigated during the last two decades, but it is still a challenging task. In this article, we provide a comprehensive survey of all ontology integration aspects. We discuss related notions and scrutinize existing techniques and literature approaches. We also detail the role of ontology matching in the ontology integration process. Indeed, the ontology community has adopted the splitting of the ontology integration problem into *matching*, *merging* and *repairing* sub-tasks, where matching is a necessary preceding step for merging, and repairing can be included in the matching process or performed separately. Ontology matching and merging systems have become quite proficient, however the trickiest part lies in the repairing step. We also focus on the case of a holistic integration of multiple heterogeneous ontologies, which needs further exploration. Finally, we investigate challenges, open issues, and future directions of the ontology integration and matching areas.

Keywords: `Ontology Integration`, `Ontology Merging`, `Ontology Matching`, `Alignment Repair`, `Coherence Principle`, `Conservativity Principle`
2010 MSC: 00-01, 99-00

1. Introduction

Ontologies have been shown to be the best means for communicating and sharing knowledge between people and machines. They provide a common understanding of a given domain thanks to their shared vocabulary (*i.e.*, terms with unified meanings). They can be used (*i*) as a query model for data sources, or (*ii*) as a basis for the integration of heterogeneous resources, *e.g.*, Web pages, XML documents, and relational databases, *etc* [1, 2].

However, ontologies do not completely solve the data integration problem. Indeed, one cannot expect all organizations to agree on the use of a common ontology. Therefore, it is unlikely that a global ontology covering all distributed systems could be developed. For instance, different applications, that have annotated their data with syntactically different but similar ontologies, work independently and cannot communicate, inter-operate or interact with each other. In fact, ontologies may cover the same domain or similar domains with different delimitations, perspectives/viewpoints, conceptual designs, granularity (levels of detail), and naming conventions. Overall, several competing, incomplete and overlapping ontologies could cover a given domain; so heterogeneity issues are inevitable. These issues arise because ontology building can never be a deterministic process in which different ontology developers make similar designing decisions. In practice, ontologies are developed independently of each other, in multiple places, by different communities and designers, for different applications, and with different requirements, prerequisites, and tools.

To leverage their power, ontologies need to overcome this semantic heterogeneity issue by integrating their distributed knowledge [3]. An ontology integration process generates a coherent integrated ontology from multiple

*Corresponding author
Email addresses: `ines.osman@fst.utm.tn` (Inès Osman), `sadok.ben@taltech.ee` (Sadok Ben Yahia), `gayo.diallo@u-bordeaux.fr` (Gayo Diallo)

input ontologies. It is an ontology reuse process [4–6] where ontologies can be reused globally or partially. Ontology integration in Big Data is impossible to achieve due to an extensive amount of data. However, integrating ontologies is strongly considered in ontology development tasks (*i.e.* when building a new ontology), since building ontologies from scratch is a labor-intensive and costly task. For example, ontology integration is useful for building application-specific ontologies (*e.g.*, ontologies of chatbot applications), or for building generic domain ontologies that can be extended or customized for specific applications. Here are some examples where it is necessary to integrate ontologies [7]:

1. Private companies may be interested in using community standard ontologies together with company-specific ontologies;
2. Collaborative companies may not only share physical assets, but may also share knowledge that needs to be integrated;
3. Companies may need to update their current ontology by adding new knowledge [8] because of new business processes requirements;
4. Applications that rely on domain ontologies may need to use ontologies covering different perspectives on one domain;
5. Ontology developers may be interested in reusing already existing ontologies as the basis for the development of new ontologies for broader domains [9]. They extend or combine them with other sub-domain ontologies.

It is worth noting that *(database) schema integration* and *ontology integration* areas have much in common [10]. Their main approaches usually comprise two primary steps: *(i)* first, the *matching* step which reconciles the differences by identifying semantic correspondences (mainly similarities) between different elements, using a similarity computation; then *(ii)* the *merging* step, which exploits the result of the matching step by merging or linking the matched elements, to produce a new unified view. An optional *repairing* step can be either included during the matching step, or be a standalone third step. Therefore, ontology matching is a necessary internal phase for ontology integration; and the improvement of ontology matching results will considerably improve ontology integration results. Automated ontology matching systems have become very efficient in the discovery of simple *equivalence* correspondences between named entities, especially between concepts. Despite this progress, there are still many issues and challenges to face in the ontology integration area. We have reported many surveys on ontology matching and its different existing approaches in the literature, *e.g.*, [11–19], *etc*. However, there is a huge lack of literature surveys dedicated to ontology integration.

In this paper, we are interested in reviewing the topic of ontology integration, and some associated aspects belonging to the ontology matching area. More precisely, the paper makes a thorough literature review regarding the different notions, approaches, issues and future avenues of the ontology integration area, with a minor emphasis on the ontology matching sub-area which plays an important role in the ontology integration process.

The remainder of the paper is structured as follows. Section 2 recalls some background knowledge about ontology, including *ontology* definitions and the *OWL* Description Logics-based language. Section 3 describes frequently used notions that are closely related to the topic of ontology integration, such as ontology *matching*, *alignment*, *mapping*, *correspondence*, and *relation*. After defining the key notions of *ontology integration* and *ontology merging* encountered in the literature, section 4 describes their different existing techniques. Section 5 explains the principles of ontology integration, the reasons behind the emergence of some issues in an integrated ontology, and the different strategies to their resolution. Section 6 briefly reviews the related work on ontology integration and merging, and sheds light on some initial observations. Section 7 gathers all the evaluation metrics used in the state of the art. Section 8 introduces two ontology integration algorithms, uses them to perform a holistic integration of multiple ontologies, then discusses the results and highlights the corresponding difficulties and challenges. Section 9 derives some interesting findings and sketches future directions from this survey. Finally, section 10 concludes the paper with a short summary.

2. Background: Ontology & OWL

According to Studer *et al.* [20], *"an ontology is a formal, explicit specification of a shared conceptualization (of a domain of discourse)"*. It is a set of triplets <$entity_1$, *relation*, $entity_2$> (*a.k.a.* <*subject*, *predicate*, *object*>). It can be viewed as a labelled directed graph whose *nodes* are entities, and *edges* are relations. Nodes are labelled by entity names, and edges are labelled by relation names. There are seven types of entities in an ontology: *concepts* (sets of individuals, or semantic categories), *object properties* (relationships or associations), *datatype properties* (or attributes),

annotation properties, *individuals* (instances, or objects), *datatypes*, and *data values* (or *data literals*). Entities are the union of all the latter ones [17]. However, in general, the are four main ontology entities, which are *concepts*, *object properties*, *datatype properties*, and *individuals*. Entities can be *named* entities or *anonymous* entities. Named entities are entities that are clearly identifiable by name, *e.g.*, a concept "*Person*". However, anonymous entities are not identifiable since they do not have a name. They are generally complex expressions, such as class or property restrictions, used in the description of named entities. For example, an anonymous class *A* can be the union/intersection of three particular named classes; To assert that a given named class *B* is disjoint to the union/intersection of the three named classes, it should be declared that the named class *B* is disjoint to the anonymous class *A*.

In RDF-based languages such as RDF (Resource Description Framework) [21, 22], RDFS (RDF Schema) [23] and OWL (Web Ontology Language) [24, 25], a concept is called a "class" or a "class of individuals". A concept can instantiate zero or more individuals; therefore, individuals are called the members of the concept or the *extension* of the concept. An individual is instantiated by a class using the built-in *"type"* relation. This instantiation is called a *class assertion*. These individuals can be linked through the object and datatype properties that are defined in the ontology. The domain of a property is its subject, and the range of a property is its object (*domain* ← *property* → *range*). An object property links an instance of a given class/*domain* to an instance of a given class/*range*. This is called an *object property assertion*. While, a datatype property links an instance of a given class/*domain* to a data value of a given datatype/*range*, such as string, integer, real, boolean, *etc*. This is called a *datatype property assertion*. Finally, annotation properties are used to annotate the whole ontology or entities of the ontology with a human-readable description, such as *comments*, entity *labels*, and ontology *version information*, *etc*.

Concepts and properties are generally organized within hierarchies since they are linked by the built-in *subsumption* relation. The latter is a taxonomic, *"is-a"*, parent/child, hypernymy/hyponymy, super-entity/sub-entity, super-type/sub-type, generalization/specialization, or broader/narrower relation. It can also be called a partial order, inheritance, or general concept inclusion (GCI) relation. Thus, three hierarchies can exist in an ontology: a class hierarchy, an object property hierarchy, and a datatype property hierarchy. The class hierarchy can be viewed as a tree-based model [26]. The latter forms a rooted acyclic graph structure [27] due to the *subsumption* relation that makes each concept subsumed by a single direct super-concept/parent (except the root concept *owl:Thing*). Nevertheless, in the case of multiple inheritance, an ontology becomes a network-based model that can contain multiple *is-a* paths leading to the same concept [26]. There are many other built-in relations that can link classes, properties or instances, such as *equivalence* and *disjointness* relations, *etc* (*See* Subsection 3.5).

In the abstract syntax, an ontology is a set of logical and non-logical axioms (*i.e.*, rules or constraints) that describe and express the entities of a domain of discourse and their associated declarations and assertions. Conceptual elements (*i.e.*, classes and properties) and the axioms that restrict their interpretation are called the ontology *schema*, *structure*, *intensional level*, or *T-Box* (for Terminological Box). Whereas, individuals and their axioms (*facts*), that instantiate the T-Box information, are called the ontology *data*, *instance data*, *extensional level*, or *A-Box* (for Assertional Box).

An IRI (Internationalized Resource Identifier) is considered as a unique name that identifies an ontology or an entity. The *full IRI*, *a.k.a.* the *prefixed name*, of an entity is composed of a *prefix* and a *suffix* separated by the symbol "#" (or by the symbol "/" or ":"). The entity prefix is usually the IRI of the ontology in which the entity appears (*e.g.*, the IRI of the current ontology, or the IRI of another existing ontology). The entity *suffix* is the local, abbreviated, or short name of the entity.

Entity IRI/Name = IRI Prefix+ "#" + IRI Suffix
Entity IRI/Name = Ontology IRI + "#" + Entity Short Name

The Web Ontology Language (OWL) [28] is an XML-based language for modeling and expressing ontologies. It is the most widely used knowledge representation language in the framework of the Semantic Web [29, 30]. OWL became a World Wide Web Consortium (W3C)[1] recommendation in 2004; and OWL 2 [25], which is a richer version of OWL, became a W3C recommendation in 2009. Compared to other XML-based languages such as RDF and RDFS, OWL endows machines with a greater ability to interpret Web content thanks to its rich vocabulary and underlying formal semantics of Description Logics. Description Logics (DL) [31] are logics that are specifically designed to represent and reason on structured knowledge. They can be considered as decidable fragments of First-Order logic (FOL). Therefore, ontologies are actually logical theories.

[1] https://www.w3.org

3. Key Notions Related to Ontology Matching

The research area of ontology integration features many terms, such as *matching*, *alignment*, *mapping*, *correspondence*, *relation*, *etc.*, which are very ambiguous and sometimes misused. Therefore, it is of utmost importance to provide a consensual definition for each term. In the following, we will refer to these terms as defined in the remainder of this section.

3.1. Ontology Matching

In general, *ontology matching*—also called *ontology alignment* although it is not preferred—is the process of discovering semantic correspondences between entities coming from different ontologies, and then generating an alignment. This process takes as input a set of N ontologies to match, denoted by Ω, and returns a set of correspondences which constitutes an alignment $\mathcal{A}$. According to Euzenat and Shvaiko [17], the ontology matching process can be seen as a function $f(\Omega, \mathcal{A}, p, r) = \mathcal{A}'$ which, from a set of ontologies Ω, an input alignment $\mathcal{A}$, a set of parameters p and a set of resources r, returns an alignment $\mathcal{A}'$ between entities of the input ontologies. An algorithm that performs an ontology matching process is called an *ontology matcher*.

In the ontology matching area, the matched entities are usually classes and properties, which is precisely referred to as *schema matching* or *T-Box matching*. However, when it comes to matching individuals, it is rather about *instance matching*, *object matching*, *coreference resolution* or *link discovery*. Ontology matching can either refer to matching an entire ontology (*i.e.*, the T-Box and the A-Box) or just the T-box of an ontology; but it generally refers to matching the T-Box [32]. We should note that most of the ontology matching solutions typically rely much more on schema-level information than on instance-level information or both schema- and level- information [17].

In the following, we will categorize different ontology matching types, and describe each one separately.

3.1.1. Simple vs. Complex Ontology Matching

The complexity/expressiveness of the resulting correspondences generates two types of ontology matching: the *simple ontology matching* for generating simple correspondences; and the *complex ontology matching* for generating complex correspondences.

- **Simple Ontology Matching** is restricted to matching named entities, which are entities identified by IRIs/namespaces, between ontologies. It is also called *one-to-one* ontology matching because, in case of matching two ontologies, it matches one named entity from the *source* ontology to one named entity from the *target* ontology ($1 : 1$). An exhaustive matching approach would compare every named entity in the first ontology to every named entity in the second ontology [32].

- **Complex Ontology Matching** extends the simple ontology matching by also matching anonymous entities, which are entities in a term building expression or formulas. That is, it can find correspondences between complex expressions. The latter are new entities constructed by named entities using either logical constructors/connectors of a logical language (DL or FOL), *e.g.* property restrictions, or using transformation functions of datatype values such as operations on strings (*e.g.*, a string concatenation), or using arithmetic operations such as a conversion [33, 34]. The complex matching is also referred to as *one-to-many*, *many-to-one*, or *many-to-many* ontology matching. Indeed, in case of matching two ontologies—which is the most common case,

  - it can match one named entity from the first ontology to an expression (*i.e.*, a construction of named entities, or a logical formulation of named entities) from the second ontology ($1 : n$);
  - it can match an expression from the first ontology to one named entity from the second ontology ($m : 1$); or
  - it can match an expression from the first ontology to an expression from the second ontology ($m : n$).

  An exhaustive matching approach would at least compare every named entity in one ontology to all possible combinations of n or m named entities in the other ontology [32].

3.1.2. Pairwise vs. Holistic Ontology Matching

The number of input ontologies to be matched generates two types of ontology matching: the *pairwise* or *binary ontology matching* [35] for matching two input ontologies; and the *holistic* or *N-ary ontology matching*, *a.k.a.* the *multiple ontology matching* [17], for matching several input ontologies.

- **Pairwise Ontology Matching** matches a *source* ontology to a *target* ontology. It takes as input a pair of ontologies to match $\Omega = \{O_1, O_2\}$ and returns an alignment $\mathcal{A}$ between these two ontologies. A matching approach needs to handle the search space between named and/or anonymous entities of only two ontologies [32].

- **Holistic Ontology Matching** generalizes the pairwise ontology matching by simultaneously inspecting multiple ontologies within the matching process. It takes as input a set of ontologies $\Omega = \{O_1, O_2, \ldots, O_N\}$ with $N > 2$, processes all input ontologies at once (in one run) and returns an alignment $\mathcal{A}$ between these ontologies [36]. A matching approach needs to handle a much larger search space between named and/or anonymous entities of N ontologies instead of two [32]. To facilitate this task, a holistic matching can rely on a set of pairwise alignments that are obtained by a pairwise matching. State-of-the-art works have implemented the two following more simple approaches:

  - **N-way Ontology Matching** [35] also takes as input a set of ontologies $\Omega = \{O_1, O_2, \ldots, O_N\}$ with $N > 2$, and returns an alignment $\mathcal{A}$ between all these ontologies. However, it implements a series of 2-way/pairwise ontology matching, by incrementally aggregating the two current matched ontologies.
  - $\binom{\mathbf{N}}{\mathbf{2}}$**-way Ontology Matching** also takes as input a set of ontologies $\Omega = \{O_1, O_2, \ldots, O_N\}$ with $N > 2$, and returns an alignment $\mathcal{A}$. However, it implements a series of pairwise matching on all the available pairs of ontologies and aggregates all their resulting alignments. For example, if $\Omega = \{O_1, O_2, O_3\}$, then the alignment $\mathcal{A}$ can be defined as $\mathcal{A} = \mathcal{A}_{12} \cup \mathcal{A}_{13} \cup \mathcal{A}_{23}$. This approach will separately make $\binom{N}{2}$ pairwise ontology matching runs [37], which corresponds to the complete pairwise alignment space between all input ontologies.

3.1.3. Oriented vs. Non-Oriented Ontology Matching

In the pairwise ontology matching case, the direction, in which the two input ontologies are matched, generates two types of ontology matching: the *oriented ontology matching* for performing a unidirectional matching from a *source* ontology to a *target* ontology; and the *non-oriented ontology matching*, which is the most common, for performing a bidirectional matching, applicable in both directions, *i.e.* from one ontology to the other, and vice versa [2, 11].

- **Oriented Ontology Matching** takes as input two ontologies and returns an oriented alignment containing "one-way" correspondences all oriented in the same direction.

- **Non-Oriented Ontology Matching** takes as input two ontologies and returns a non-oriented alignment containing "two-way" correspondences oriented in both directions.

3.1.4. Compound vs. Ternary Ontology Matching

When it is about a holistic complex matching, the origin of the entities that are composing a complex expression of a given correspondence generates two types of ontology matching: the *compound ontology matching* for matching expressions that involve a combination of entities originating from two or more input ontologies; and the *ternary compound ontology matching* for matching a *source* ontology to two *target* ontologies [38, 39], which is a special case of a compound matching.

- **Compound Ontology Matching** [38] takes as input at least three ontologies to match $\Omega = \{O_1, O_2, \ldots, O_N\}$ with $N > 2$, and returns a compound alignment $\mathcal{A}$. It can discover correspondences involving complex expressions that are constructed by existing named entities coming from two or more input ontologies. It needs to handle a large search space between entities of N ontologies, and also be able to compose expressions using entities from different ontologies. An exhaustive matching approach would at least compare every named entity in one ontology to all possible combinations of named entities in the other $(N - 1)$ ontologies [32].

- **Ternary Ontology Matching** [38] takes as input three ontologies to match $\Omega = \{O_1, O_2, O_3\}$, and returns a ternary compound alignment $\mathcal{A}$ containing *equivalence* correspondences. It matches named entities from the first input ontology to expressions involving named entities from the second and third input ontologies. It needs to handle a search space between entities of three ontologies, and also be able to compose expressions using entities from two different ontologies. An exhaustive matching approach would compare every named entity in the first ontology to all possible combinations of named entities in the other two ontologies [32]. This is the simplest case of a compound matching.

In the literature, a *simple pairwise non-oriented* matching is called, in short, an *ontology matching*. In the complexity scale, we categorize ontology matching levels in an ascending order as follows: *simple*, *complex*, *ternary*, and finally *compound* matching.

3.2. Ontology Alignment

A semantic alignment is a set of semantic correspondences between two or more matched ontologies, denoted by $\mathcal{A} = \{c_1, c_2, \ldots, c_n\}$, and stored separately. It is the result of the ontology matching process. Alignments can be used in various tasks, such as query response or distributed querying, query rewriting, logical reasoning on ontologies, data interlinking or Semantic Web browsing, ontology transformation, and ontology integration & merging. Following the aforementioned ontology matching types, we distinguish eight alignment types:

- **Simple Alignment** only contains simple entity-to-entity correspondences. The most used format for representing simple pairwise alignments is the *RDF Alignment* format[2] [40], *a.k.a.* the *Alignment format* or the *Alignment API format*—where Alignment is written with a capitalized "A". This format is expressed in RDF. It is the most consensual ontology alignment format, despite its simplicity and its lack of expressiveness and thus, inability to represent all kinds of correspondences.

- **Complex Alignment** contains at least one complex correspondence—expressed as a logical rule. In this case, *EDOAL*[3] (Expressive and Declarative Ontology Alignment Language) [40] is the most used format for representing complex pairwise alignments. It extends the Alignment format by also allowing to express complex correspondences.

- **Pairwise Alignment**, *a.k.a. binary alignment* [35], only involves two ontologies to match: a *source* ontology and a *target* ontology; and only contains correspondences linking two entities, each coming from an ontology.

- **Holistic Alignment**, *a.k.a. multi-alignment* or *multialignment* [17], involves more than two ontologies and contains at least one correspondence that links different entities coming from more than two input ontologies.

- **Oriented Alignment**, *a.k.a. mapping* [17] or *directional alignment* [41], is the directed or oriented version of a pairwise alignment [17], denoted $\mathcal{A}_{O_1 \to O_2}$ where the *source* ontology is O_1 and the *target* ontology is O_2. Correspondences are total functions rather than relations [2]. When an alignment is unidirectional, $\mathcal{A}_{O_1 \to O_2}$ is not the inverse of $\mathcal{A}_{O_2 \to O_1}$ [41].

- **Non-Oriented Alignment** is a non-directed or non-oriented pairwise alignment, denoted $\mathcal{A}$ or $\mathcal{A}_{O_1 \leftrightarrow O_2}$ and briefly called an *alignment*. When an alignment is a bidirectional one, $\mathcal{A}_{O_1 \leftrightarrow O_2}$ is the inverse of $\mathcal{A}_{O_2 \leftrightarrow O_1}$. Since a binary relation can be decomposed into a pair of total functions, the bidirectional alignment can be considered as a pair of oriented alignments, so $\mathcal{A}_{O_1 \leftrightarrow O_2} = \mathcal{A}_{O_1 \to O_2} + \mathcal{A}_{O_2 \to O_1}$ [11].

- **Compound Alignment** involves more than two ontologies, and contains at least one complex correspondence that involves an expression composed of named entities from two or more ontologies.

- **Ternary Alignment** is a special case of a compound alignment. It involves three ontologies: a *source* and two *targets*; and contains at least one complex *equivalence* correspondence between a named entity from the *source* ontology and an expression involving two named entities from the two *target* ontologies.

In the literature, a *simple pairwise non-oriented* ontology alignment is called, in short, an *alignment*.

[2] http://alignapi.gforge.inria.fr/format.html
[3] http://alignapi.gforge.inria.fr/edoal.html

3.2.1. Alignment Cardinality/Multiplicity

The notion of *alignment multiplicity* is often used in the case of two matched ontologies. It describes how many entities from one ontology can be matched to an entity from another ontology [17]. Alignments can have different cardinalities. Usual notations are *one-to-one* $(1:1)$, *one-to-many* $((1:m)$ or $(1:*))$, *many-to-one* $((n:1)$ or $(*:1))$, and *many-to-many* $((n:m)$ or $(*:*))$, where $*$ means zero or more entities. Alignments are usually *partial* alignments (*i.e.*, not *total* alignments); that is there could be many entities in both ontologies that have no correspondence in the other ontology. As a result, alignments' cardinalities can rather have the following notations: $(?:?)$, $(?:*)$, $(*:?)$ and $(*:*)$, where ? means zero or one entity. In general, alignments between independently developed ontologies are *many-to-many* alignments, where zero or more entities from the first ontology can match with zero or more entities from the second ontology.

Ambiguous Alignment [17] contains some correspondences that share a common entity: It matches the same entity from one ontology with several entities from the other ontology. In other words, an ambiguous pairwise alignment contains multiple correspondences having in common a *source* entity or a *target* entity. Therefore, alignments of cardinalities $(1:*)$ or $(1:m)$, $(*:1)$ or $(n:1)$, and $(*:*)$ or $(n:m)$ are actually ambiguous alignments. In contrast, in *one-to-one* alignments $(1:1)$, an entity (*i.e.*, a *source* or *target* entity) appears in at most one correspondence.

3.3. Alignment Correspondence

A semantic correspondence is also called a *mapping*, a *match*, a *relationship* or a *relation* by some authors, although these terms are not preferred. Commonly, given two matched ontologies O_1 and O_2, a correspondence is a 5-tuple $< id, e_{O_1}, e_{O_2}, r, n >$ [17] that can be denoted $< e_{O_1}\ r\ e_{O_2} >$, such that:

- *id* is the unique identifier of the correspondence.
- e_{O_1} and e_{O_2} are the members of the correspondence. e_{O_1} is an entity belonging to O_1, and e_{O_2} is an entity belonging to O_2.
- r is a binary semantic relation holding or intended to hold between e_{O_1} and e_{O_2}, such as *equivalence* $(\equiv)$, *subsumption* $(\sqsubseteq/\sqsupseteq)$, *disjointness* $(\perp)$, *instantiation*, *overlap* $(\between)$, or named relations, *etc*.
- n is a confidence measure assigning a degree of trust on the identified relation. It is a real number that is generally normalized in the interval $[0, 1]$ to reflect the degree of truth/correctness/reliability of the correspondence. In the equivalence case, n indicates whether the two entities have a high or low similarity degree. The correspondence asserts that the relation r links e_{O_1} and e_{O_2} with a confidence value equal to n and ranging between $[0, 1]$. The higher the confidence degree, the more likely the relation holds [17].

In the *crisp* alignments, the confidence values of all correspondences are equal to 1.0. Following the preceding matching types, we distinguish eight correspondence types:

- **Simple Correspondence** involves a named entity in all of its members. A named entity can be a named concept, a named property, or a named individual. For example, if it is a pairwise correspondence, e_{O_1} and e_{O_2} are simple named entities. The following example is a simple pairwise correspondence written in DL and FOL languages, respectively:

$$O_1\texttt{:Person} \equiv O_2\texttt{:Human}$$
$$\forall \texttt{x},\ O_1\texttt{:Person(x)} \equiv O_2\texttt{:Human(x)}$$

- **Complex Correspondence** involves an anonymous entity in at least one of its members, *e.g.* in e_{O_1} or e_{O_2} or both, if it is a pairwise correspondence. An anonymous entity is constructed by a logical formulation. The following example is a complex pairwise correspondence written in DL and FOL languages, respectively:

$$O_1\texttt{:Accepted\_Paper} \equiv O_2\texttt{:Paper} \sqcap \exists\ O_2\texttt{:hasDecision.Acceptance}$$
$$\forall \texttt{x},\ O_1\texttt{:Accepted\_Paper(x)} \equiv \exists \texttt{y},\ O_2\texttt{:Paper(x)} \wedge O_2\texttt{:hasDecision(x,y)} \wedge O_2\texttt{:Acceptance(y)}$$

- **Pairwise Correspondence** is a binary correspondence between two ontologies, denoted as $c = < e_{O_1}, e_{O_2} >$, such that e_{O_1} belongs to O_1 and e_{O_2} belongs to O_2. The following example [32] is a complex pairwise correspondence written in DL and FOL languages, respectively:

$$O_1\texttt{:Professor} \equiv O_2\texttt{:AsstProf} \sqcup O_2\texttt{:AssocProf} \sqcup O_2\texttt{:FullProf}$$
$$\forall \texttt{x},\ O_1\texttt{:Professor(x)} \equiv O_2\texttt{:AsstProf(x)} \vee O_2\texttt{:AssocProf(x)} \vee O_2\texttt{:FullProf(x)}$$

- **Holistic Correspondence**, also called *N-ary correspondence* [17] or *clique* [36], stands for a correspondence between multiple ontologies, denoted as $c = < e_{O_1}, e_{O_2}, \ldots, e_{O_N} >$, such that each e_{O_j} belongs to an ontology O_j and $N > 2$. It can only be an *equivalence* or *disjointness* correspondence. In the sake of completeness of the alignment, it could be observed that the holistic correspondence does not need to necessarily contain an entity from each ontology. Some components (e_{O_j}) of the holistic correspondence may be empty. In other terms, a holistic correspondence should be composed of at least two entities, not necessarily exactly N entities. Because when trying to only find correspondences composed of N entities, other correspondences that are not shared among all input ontologies can be missed. The following example [36] is a simple holistic correspondence expressed in DL and FOL languages, respectively:

$$O_1\texttt{:writePaper} \equiv O_2\texttt{:writes} \equiv O_3\texttt{:hasRelatedPaper}$$
$$\forall \texttt{x},\ O_1\texttt{:writePaper(x)} \equiv O_2\texttt{:writes(x)} \equiv O_3\texttt{:hasRelatedPaper(x)}$$

- **Oriented/Directional Correspondence** is a directed or ordered correspondence oriented from a *source* ontology to a *target* ontology. It is a one-way or injective function [2] relating two entities, such as *subsumption* relations or named (not built-in) relations. In a directional correspondence, $< e_{O_1}\ r\ e_{O_2} >$ is not the inverse of $< e_{O_2}\ r\ e_{O_1} >$ [41]. The following example is a simple oriented pairwise correspondence in DL and FOL languages, respectively:

$$O_1\texttt{:Man} \sqsubseteq O_2\texttt{:Person}$$
$$\forall \texttt{x},\ O_1\texttt{:Man(x)} \sqsubseteq O_2\texttt{:Person(x)}$$

- **Non-Oriented/Bidirectional Correspondence** is an undirected or unordered correspondence oriented in both directions: from a *source* to a *target* ontology, and vice versa. It is a two-way or bijective function [2] relating two entities, such as *equivalence* or *disjointness* relations. When a correspondence is bidirectional, $< e_{O_1}\ r\ e_{O_2} >$ is the inverse of $< e_{O_2}\ r\ e_{O_1} >$.

- **Compound Correspondence** involves a complex compound entity in at least one of its members. A complex compound entity is an entity constructed by a logical formulation using named entities coming from two or more input ontologies.

- **Ternary Correspondence** involves a named entity in one member and a complex compound entity in the other member. The named entity comes from one input ontology. The complex compound entity is a logical formulation constructed by two named entities coming from two other input ontologies. The following example [32, 39] is a ternary correspondence written in DL and FOL languages:

$$O_1\texttt{:aorticStenosis} \equiv O_2\texttt{:aorta} \sqcap O_3\texttt{:constricted}$$
$$\forall \texttt{x},\ O_1\texttt{:aorticStenosis(x)} \equiv O_2\texttt{:aorta(x)} \wedge O_3\texttt{:constricted(x)}$$

In the literature, a *simple pairwise bidirectional* alignment correspondence is briefly called a *correspondence*. Table 1 summarizes different types of ontology matching, ontology alignment, and alignment correspondence.

Table 1: Summary of *Matching*, *Alignment* and *Correspondence* Types

	Ontology Matching Type (Subsection 3.1)	**Ontology Alignment Type** (Subsection 3.2)	**Alignment Correspondence Type** (Subsection 3.3)
Subsection 3.1.1	**Simple Matching** A simple matching produces a simple alignment.	**Simple Alignment** A simple alignment is composed of a set of simple correspondences.	**Simple (or 1-to-1) Correspondence** A simple correspondence links named entities, each coming from an input ontology. (*See* also Subsection 3.3.2)
	Complex Matching A complex matching produces a complex alignment.	**Complex Alignment** A complex alignment is composed of a set of complex correspondences.	**Complex (or *n*-to-*m*) Correspondence** A complex correspondence links entities such that at least one of them is an anonymous entity (often coming from an input ontology). (*See* also Subsection 3.3.2)
Subsection 3.1.2	**Pairwise Matching** A pairwise matching produces a pairwise alignment.	**Pairwise Alignment** A pairwise alignment is composed of a set of pairwise correspondences.	**Pairwise Correspondence** A pairwise correspondence links only two entities, each coming from an input ontology: It links an entity from a first (*source*) ontology to an entity from a second (*target*) ontology.
	Holistic Matching A holistic matching produces a holistic alignment.	**Holistic Alignment** A holistic alignment is composed of a set of holistic correspondences.	**Holistic Correspondence** A holistic correspondence links many entities, each coming from an input ontology.
Subsection 3.1.3	**Oriented Matching** An oriented matching produces an oriented alignment.	**Oriented Alignment** An oriented alignment is composed of a set of oriented correspondences.	**Oriented Correspondence** An oriented correspondence is a pairwise correspondence directed from a *source* ontology to a *target* ontology. It can be a *subsumption* relation or a named relation.
	Non-Oriented Matching A non-oriented matching produces a non-oriented alignment.	**Non-Oriented Alignment** A non-oriented alignment is composed of a set of non-oriented correspondences.	**Non-Oriented Correspondence** A non-oriented correspondence is a pairwise correspondence oriented in both directions (*i.e.*, bidirectional). It can be an *equivalence* or *disjointness* relation.
Subsection 3.1.4	**Compound Matching** A compound matching produces a compound alignment.	**Compound Alignment** A compound alignment is composed of a set of compound correspondences.	**Compound Correspondence** A compound correspondence is a complex correspondence in which there is at least one anonymous entity constructed by named entities coming from two or more input ontologies.
	Ternary Matching A ternary matching produces a ternary alignment. It is a special case of a compound matching.	**Ternary Alignment** A ternary alignment is composed of a set of ternary correspondences.	**Ternary Correspondence** A ternary correspondence is a complex correspondence that links a named entity coming from the first (*source*) ontology to an anonymous entity constructed by two named entities coming from the second and third (*target*) ontologies.
—	**(Ambiguous) Matching** A matching of two independently developed ontologies often produces an ambiguous alignment.	**Ambiguous (or *n*-to-*m*) Alignment** An ambiguous alignment contains a set of ambiguous correspondences. (Subsection 3.2.1)	**Ambiguous Correspondence** An ambiguous correspondence is a pairwise correspondence in which at least one entity appears in (one or more) other correspondences. (Subsection 3.3.1)
	(Non-Ambiguous) Matching A matching of two very similar ontologies often produces a non-ambiguous alignment.	**Non-Ambiguous (or 1-to-1) Alignment** A one-to-one alignment is composed of non-ambiguous correspondences. It does not contain ambiguous correspondences.	**Non-Ambiguous Correspondence** A non-ambiguous correspondence is a pairwise correspondence in which no entity appears in other correspondences.

A *simple pairwise non-oriented* matching, alignment and correspondence is often briefly called a "matching", an "alignment" and a "correspondence", respectively.

3.3.1. Ambiguous Correspondence

Following the preceding definition of an ambiguous alignment, we introduce the definition of an *ambiguous correspondence* [17], *a.k.a.* a *correspondence of higher multiplicity* [42] or a *higher-multiplicity correspondence* [42]. It is a correspondence in which at least one member (*i.e.*, one entity) is also involved in other correspondences. In the ambiguous pairwise alignment case, a *source* or *target* entity occurs in at least two correspondences. Ambiguous pairwise correspondences match the same entity from a first ontology with more than one entity from a second ontology. The following example shows three ambiguous correspondences in DL:

$$\begin{aligned} O_1\texttt{:Student} &\equiv O_2\texttt{:Student} \\ O_1\texttt{:Student} &\equiv O_2\texttt{:Scholar} \\ O_1\texttt{:Student} &\equiv O_2\texttt{:PhD\_Student} \end{aligned}$$

3.3.2. Correspondence Cardinality/Multiplicity

The notion of *correspondence multiplicity* is different from the notion of *alignment multiplicity* (*See* Subsection 3.2.1). This notion is also often used in the case of two matched ontologies. It describes how many entities from each ontology are involved in one correspondence (not in a whole alignment). It reflects the correspondence complexity. Correspondences can have different cardinalities[4]. Usual notations are *one-to-one* $(1:1)$ for the simple correspondences; and *one-to-many* $((1:m)$ or $(1:*))$, *many-to-one* $((n:1)$ or $(*:1))$ and *many-to-many* $((n:m)$ or $(*:*))$ for the complex correspondences, where $*$ means zero or more entities.

3.4. Ontology Mapping

In the literature, the term *mapping* can correspond to either a *matching* process, an *alignment* file, or a *correspondence*. Indeed, *mapping* can be used as a noun and a verb in the "-ing" form, which can be confusing. It is better to avoid it and use instead the aforementioned terms. However, some authors try to propose a standard definition for the term *mapping*. Faria *et al.* [43] defined a mapping as an *equivalence* correspondence. Whereas Euzenat and Shvaiko [17] defined it as a non-ambiguous oriented/directed equivalence alignment, containing oriented/directional *equivalence* correspondences that can be called *mapping rules* once interpreted as ontological statements/axioms.

3.5. Correspondence Relation

A semantic relation of a given correspondence can be a built-in relation such as the *equivalence* relation ($\equiv$), the *subsumption* relation ($\sqsupseteq$ or $\sqsubseteq$), the *disjointness* or *incompatibility* relation ($\perp$), and the *instantiation* relation (*i.e.*, *has instance* or *instance of*). In an alignment, relations can be flagged by the following symbols: "=" (*i.e.* is equivalent to), ">" (*i.e.* subsumes or is more general than), "<" (*i.e.* is subsumed by or is more specific than), and "%" (*i.e.* incompatible with). A semantic relation is not only restricted to built-in/predefined relations of the OWL ontology language, but also includes other relations such as the *overlap* relation ($\between$ or $\sqcap$), named relations, and fuzzy relations. In the following, we describe the three most common built-in relations:

- **Equivalence Relation**: An equivalence relation holding between two classes A and B means that all instances of A are also instances of B, *i.e.* both classes contain exactly the same set of individuals as instances. An equivalence relation holding between two properties P_1 and P_2 means that an individual x can be connected to an individual or a data literal y by both P_1 and P_2. An equality relation holding between two individuals x and y means that the individual x is identical/equal/synonymous to the individual y and refers to the same thing.

- **Subsumption Relation**: A subsumption relation holding between two classes A and B means that the set of instances of A is a subset/superset of the set of instances of B. A subsumption relation holding between two properties P_1 and P_2 means that if an individual x is connected by P_1 to an individual or a data literal y, then x is also connected by P_2 to y.

[4]Notations of multiplicity are introducing ambiguity in the literature because multiplicity notations of both alignments and correspondences are the same. Therefore, in every mention of the word *multiplicity*, we precise whether it is an "alignment" multiplicity or a "correspondence" multiplicity.

- **Disjointness Relation**: A disjointness relation holding between two classes A and B means that instances of A are definitely not instances of B. In other words, no individual x can be at the same time, an instance of both A and B. A disjointness relation holding between two properties P_1 and P_2 means that no individual x can be connected to an individual or a data literal y by both P_1 and P_2. An inequality relation holding between two individuals x and y means that the individual x cannot be identical/equal/synonymous to y and does refer to a different thing.

According to Cheatham and Pesquita [32], the simplest types of relations are *equivalence* and *disjointness*, and then comes the *subsumption* relation. However, Solimando *et al.* [44] consider *equivalence* and *subsumption* to be the simplest relations, and then comes the *disjointness* relation, because negative constraints are typically harder to identify and assess than positive ones. That is why most of the available matching systems do not compute *disjointness* relations. Arbitrary relations are even harder to find. An arbitrary relation [32] is any type of relations: *equivalence*, *subsumption*, *disjointness*, *etc*. Consequently, the simplest type of ontology matching is the one that finds simple pairwise *equivalence* correspondences, which is the most common ontology matching type. Then, the one that finds simple pairwise *subsumption* and *disjointness* correspondences comes next in the difficulty level.

3.6. Network of Ontologies

Denoted as $< \Omega, \Lambda >$, a *network of ontologies* [17], *a.k.a.* a *network of aligned ontologies* [45] or *networked ontologies* [46], is composed of a set of ontologies Ω and a set of pairwise alignments Λ between arbitrary pairs of these ontologies. It constitutes a sort of *distributed ontology* [47].

A ***normalized network of ontologies*** [17], denoted as $< \Omega, \Lambda >$, is a network of ontologies with exactly one alignment between each possible pair of its ontologies. In other words, a network of ontologies is called a *normalized* one, if and only if $|\Lambda(O, O')| = 1$ for any two ontologies O and O' composing it, where $\Lambda(O, O')$ denotes the set of existing alignments between O and O'.

4. Ontology *Integration*

In this section, we provide some of the consensual definitions of the notions of *ontology integration* and *ontology merging* in the literature and describe their different existing types.

4.1. Ontology Integration vs. Ontology Merging

Ontology *integration* and *merging* terms are ambiguous. Some authors consider and use both of them as synonyms, whereas other authors do not. In the remainder, we will refer to these terms as defined in this subsection.

In general, ontology integration/merging is the process of integrating/merging two or more ontologies into a single one [48]. However, more precisely, ontology *integration* or *merging* is the process of reusing or unifying existing ontologies to build a new more general or more complete one that can be utilized by a specific application or by existing applications already using the input ontologies that were integrated or merged [4–6]. Ontology merging is sometimes referred to as *ontology fusion* [4–6, 49].

Euzenat and Shvaiko [17] defined ontology merging by the process of creating a new ontology from two or more, possibly overlapping, source ontologies. As for the integration, they stated that ontology integration is the inclusion, in an ontology, of another ontology along with links between these two ontologies, as prescribed in the alignment between them. The only difference between these two definitions is that, unlike ontology merging which does not alter the input ontologies in the merging process, ontology integration alters one of the input ontologies—the *target* ontology—and keeps the other *source* ontologies unaltered. The authors [17] formalized the merging of two ontologies by a merge operator, $Merge(O_1, O_2, \mathcal{A}) = O_3$, where O_1 and O_2 are the source ontologies to be merged, and $\mathcal{A}$ is the alignment expressed in the same logical language as ontologies O_1 and O_2. As for the ontology integration, we suggest the integration operator, $Integrate(O_1, O_2, \mathcal{A}) = O_1$, where O_1 is the *target* ontology into which the *source* ontology O_2 will be integrated.

We believe that the ontology *integration* notion is synonymous to the ontology *enrichment*, *inclusion* or *extension* notions, and that it also includes the *merging* notion. In other words, we consider merging as a special case of integration. In fact, the result of including/integrating *source* ontologies into an empty *target* ontology is equivalent to the result

of merging them. For example, if it is about two input ontologies O_1 and O_2, the resulting ontology will be O_1 in the integration case, and will be O_3 in the merging case. Therefore, merging ontologies can be considered as integrating ontologies into one another. More simply, merging two ontologies O_1 and O_2 can be considered as simultaneously integrating O_1 into O_2 and O_2 into O_1 (in both directions). Consequently, a resulting *merged* ontology can be called an *integrated* ontology. Hence, we have chosen to use the term *integration* in the title and all the remaining, rather than *merging*. In Table 2, we summarize the differences between definitions of *ontology integration* and *ontology merging*.

Table 2: Description of *Ontology Merging* and *Ontology Integration*

	Ontology Merging		**Ontology Integration**		
Integration Type	Simple Merge, or Full Merge		Simple Merge, Full Merge, or Asymmetric Merge		
Final Goal	To build a brand new ontology		To enrich and extend an ontology		
Method	Assembling/Aggregation		Inclusion		
Method Details	Assembling *source* ontologies to form a *target* ontology		Incorporating *source* ontologies -one by one- into a *target* ontology		
Linkage*	Weak or strong		Weak or strong		
Input Ontologies	2 or more *source* ontologies		1 *target* ontology + 1 or more *source* ontologies		
Output Ontology	A newly built *target* ontology, called a "merged" ontology		A new version of the *target* ontology, called an "integrated" ontology		
Results	The *source* ontologies are replaced by the "merged" ontology		The *target* ontology is replaced/overwritten by the "integrated" ontology		
Integration Purpose	To interoperate applications already using the *source* ontologies	To build an ontology in order to be used by a particular application	To enrich an ontology already used by existing applications	To enrich an ontology under construction, to be used by a particular application	To enrich an empty ontology, which leads to *ontology merging*
***Source* Ontologies' Knowledge Preservation**	Mandatory	Not mandatory	Not mandatory	Not mandatory	*See* the ontology merging column
***Target* Ontology's Knowledge Preservation**	–	–	Mandatory	Not mandatory	*See* the ontology merging column
Domains of the Input Ontologies	Same/Similar domains	Same/Related or different domains	Same/Similar domains	Same/Related or different domains	*See* the ontology merging

* Weak: equivalent entities are simply linked by *equivalence* axioms.

* Strong: equivalent entities are completely merged to constitute a single entity.

4.2. Types of Ontology Integration

We distinguish three distinct ontology integration approaches derived from our extensive state-of-the-art research, namely: (*i*) the *simple merge* (or the *bridge ontology*) and (*ii*) the *full merge*, which are semantically equivalent; and (*iii*) the *asymmetric merge* which is actually an ontology enrichment. Only the targeted application and usage can justify preferring one integration approach over another.

4.2.1. Simple Merge

The first approach is the *Simple Merge*, *a.k.a.* the *Reduced Semantics* [17] or the *Simple Union* [27]. It imports the input ontologies into a new ontology—constituting a union of input ontologies—and adds bridge/bridging/articulation axioms translating the alignment between them (*See* Figure 1a). These added axioms are actually semantic correspondences interpreted as or transformed into ontological statements to link/connect the overlapping part of the input ontologies. They are called a *semantic bridge* [50], *articulations*, or *reductionistic alignment semantics* [51] (abbreviated as *alignment semantics* or *reductionistic semantics*). The reductionistic alignment semantics is so called, because the semantics of an alignment is reduced to the semantics of Description Logics, thus to a set of axioms. The correspondences of the alignment $\mathcal{A}$ can be perceived as an ontology $O_{\mathcal{A}}$ called *(semantic) bridge ontology* [1, 2, 52], *articulation ontology* [50], *articulation* [50], *intersection ontology* [53], or *intermediate ontology* [11]. Following the recommendation of the W3C best practices group [54], most of the state-of-the-art approaches assume that the input ontologies are in OWL and correspondences between them are also interpreted as OWL axioms:

- The *subsumption*, *equivalence*, and *disjointness* correspondences between classes are expressed by built-in *subClassOf*, *equivalentClass* and *disjointWith* OWL axioms, respectively;
- The correspondences between properties are expressed by built-in *subPropertyOf*, *equivalentProperty* and *propertyDisjointWith* OWL axioms, respectively; and
- The *equality* and *inequality* correspondences between individuals are expressed by built-in *sameAs* and *differentFrom* OWL axioms, respectively.
- Whereas the identifiers and the confidence values of the correspondences are expressed by OWL annotations, which are non-logical axioms having no effect on inferences [55].

Therefore, in the case of two input ontologies, the integrated ontology O_3 can be viewed as the union of O_1, O_2 and $O_{\mathcal{A}}$ where $O_3 = O_1 \cup O_2 \cup O_{\mathcal{A}}$ [55]. In this type of merge, equivalent entities are mentioned more than once in the integrated ontology but linked by *equivalence* axioms, hence they can be considered as non-redundant entities (*See* Figure 1a). It should be noted that correspondences can be interpreted in other languages[5] other than OWL, such as SWRL (Semantic Web Rule Language) [59] or SKOS (Simple Knowledge Organization System) [60]. The resulting ontology can be called an *aligned* ontology [44, 61], a *merged* ontology or an *integrated* ontology.

To achieve ontology modularization, the OWL language provides a built-in import statement <*owl:imports*>. The latter includes the content of an entire ontology into the current ontology by only referencing the URI or the local file of that ontology. Therefore, the integrated ontology O_3 is generally obtained by converting the RDF alignment to an OWL ontology O_A which directly imports the two input ontologies O_1 and O_2; so that O_A becomes O_3. Otherwise, the integrated ontology O_3 is obtained by creating an empty ontology that directly imports O_1, O_2 and O_A. The import is automatically performed by simply declaring the OWL import statements referencing the ontologies to be imported. In this case, the resulting ontology O_3 is generally called a *bridge ontology*.

4.2.2. Full Merge

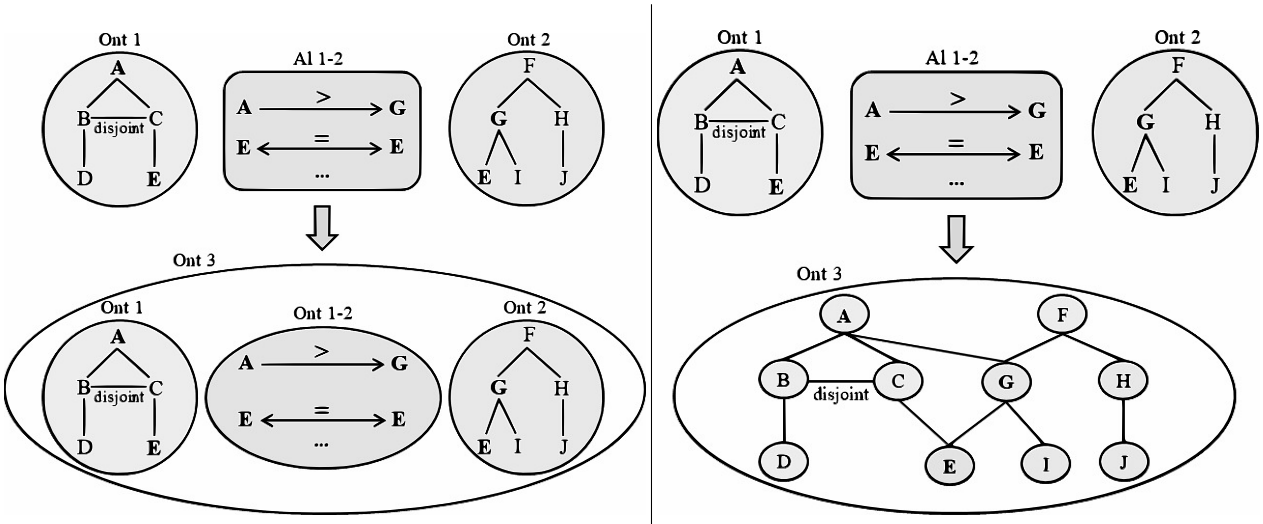
The second approach is the *Full Merge* [27], *a.k.a.* the *Complete Merge* [1, 2] or the *Symmetric Merge* [27]. It imports the input ontologies into a new ontology—constituting a union of input ontologies—and merges/combines each set of equivalent entities into a single new entity that preserves all their attached description and relations (*See* Figure 1b). Ontological axioms, constituting the merged ontology and originating from the input ontologies, are updated by replacing every occurrence of the original entities with its new merged entity. That is, each axiom, in which appears the name of one of the entities that have been merged, must be updated by replacing the name of that original entity with the name of the newly merged one. *Subsumption* axioms can also be added to link subsuming and subsumed entities, as prescribed in the alignment(s).

In the literature, authors identify the merged entities by either a unique (alphanumeric) code or by the name of one of the original entities that have been merged—commonly, by the name of the entity that belongs to the preferred

[5] Some other languages or formalisms have been proposed and used in the literature for expressing the semantic correspondences, such as C-OWL (Contextualized OWL) [56], DDL (Distributed Description Logics) [57] and ε-connected OWL [58] that are designed to reason on ontologies connected by directional alignments, *i.e.* to reason on a network of aligned ontologies, not in a merged ontology as in our case.

input ontology; then, they add the short names of the original entities (that have been merged) as additional labels to the newly merged entity. The merged entities will be represented only once, which avoids the existence of redundant entities in the merged ontology. In the case of two input ontologies, the merged ontology O_3 can be viewed as the union of O_1 and O_2 where $O_3 = O_1 \cup O_2 = (O_1 - O_2) \cup (O_2 - O_1) \cup (O_1 \cap O_2)$ [1]. Multiple inheritance, resulting from assigning more than one direct parent to each merged entity, will generate multiple root *is-a* paths to the entities of the merged ontology. The resulting ontology can be referred to as a *unified* ontology [4, 50, 62], a *merged* or an *integrated* ontology.

After choosing an input ontology as a transforming ontology, a full merge can also be performed by applying an *ontology transformation* process to $N - 1$ input ontologies, using pairwise alignments between the transforming ontology and each of the $N - 1$ ontologies to be transformed. Then, an *ontology composition/aggregation* process is applied to all of the N input ontologies (*i.e.*, the transforming ontology and the $N - 1$ transformed ones) [17]. For instance, in the case of two ontologies O_1 and O_2 and an alignment $\mathcal{A}$ between them, entities of O_2 are expressed by equivalent entities of O_1 (or inversely) as mentioned in the alignment $\mathcal{A}$, then O_1 and O_2 are aggregated to constitute the merged ontology O_3. Therefore, $O_3 = O_1 \cup O_2^{'}$ such that $O_2^{'} = Transform(O_2, \mathcal{A})$.



(a) Simple-Merge Ontology

(b) Full-Merge Ontology

Figure 1: Ontology Integration/Merging Types

4.2.3. Asymmetric Merge

The third approach is the *asymmetric merge* [63, 64], *a.k.a.* the *target-driven asymmetric merge*, the *ontology enrichment* [42] or the *ontology integration* [42]. It takes one of the input ontologies as the *target* ontology into which the other *source* ontologies will be incrementally integrated to enrich/extend it. The merging manner can follow the full merge approach or the simple merge approach.

The *target* ontology is the priority (preferred) ontology, whereas the *source* ontologies have a lower priority. The *target* ontology is also called the *seed* ontology [42], the *backbone* ontology [42] or the *knowledge base* [8, 42]. The latter should be completely preserved during the integration process because it may already be in use by various applications or services. Thus, all of its axioms (especially its structure) should not be altered, and all of its entities should be preserved. However, entities and axioms (including the structure) of the *source* ontologies can be removed or modified if they are redundant or in disagreement with the *target* ontology conceptualization. In this respect, we believe that this *asymmetric merge* is actually an *integration* or an *enrichment* of the *target* ontology using merging techniques, rather than a *merge*. The resulting ontology is called an *integrated* ontology.

5. Ontology Integration Principles, Violations and Repair

In this section, we explain the principles that an ontology integration should fulfill, the reasons behind the emergence of errors (or issues) in the integrated ontologies, and the repair strategies applied to resolve these errors.

5.1. Ontology Integration Principles (and Examples of their Violation)

Integrating ontologies using alignments can lead to unintended consequences such as semantic conflicts or contradictions, redundancies and cycles, *etc*. Pottinger and Bernstein [65] proposed a set of requirements for merging models such as database schemata, ontologies, and UML models, *etc*. However, these merge requirements are very generic and need to be specialized for the case of ontology merging (and integration). In this paper, we have gathered the most relevant requirements (or principles) that an ontology integration should try to fulfill, and they are as follows:

I. Integrated Ontology Requirements

(a) *Alignment correctness and completeness*,
(b) *Ontology knowledge preservation*,
  i. *Entities preservation*,
  ii. *Axioms/Entailments preservation*,
(c) *Alignment correspondences preservation*,
(d) *Ontology coherence (and consistency)*,
(e) *Ontology conservativity*,

II. General Ontology Requirements

(a) *Minimality* (or *prohibition of redundant entities*),
(b) *Acyclicity* (or *prohibition of subsumption cycles*),
(c) *Prohibition of structural and relational redundancy*,
(d) *Property's domain and range oneness*,
(e) *Entity connectivity* (or *prohibition of unconnected entities*), *etc*.

We distinguish the specific requirements of an integrated ontology and the general requirements of any ontology. It is impossible to simultaneously satisfy all the ontology integration requirements, but they should be satisfied as much as possible. We have added a requirement that we have called the *prohibition of structural and relational redundancy*. The latter, as well as the principles (a), (b), (d) and (e), are requirements that any ontology should preferably fulfill (not particularly an integrated one). In the following, we explain each principle/requirement in a separate subsection, and we support them with some violation examples.

5.1.1. Alignment Correctness and Completeness Principle

This feature only depends on the performance of the *ontology matching* step. Ontology matching tools may either lack correct correspondences, contain incorrect correspondences, or both. Ideally, the ontology alignment should not contain any false correspondences, which reflects the *alignment correctness*, and should contain all the correct correspondences, which reflects the *alignment completeness*.

5.1.2. Ontological Knowledge Preservation Principle

The principle of *knowledge preservation*, *a.k.a. knowledge completeness* or *knowledge coverage* [27], is the preservation of all entities and axioms from the input ontologies that were integrated. Entities or elements include all classes (*i.e.*, *class preservation*), properties (*i.e.*, *property preservation*) and individuals (*i.e.*, *instance preservation*) from the input ontologies. Axioms include all kinds of triplets such as *subsumption*, *equivalence* and *disjointness* axioms, annotations, declarations, expressions, restrictions and assertions from the input ontologies; not only *subsumption* axioms of their class hierarchies (*i.e.*, *relationship preservation* [65] or *structure preservation*). Indeed, all entailments (*i.e.*, inferred axioms) of the input ontologies should be preserved in the integrated ontology [55]. Hence, the knowledge preservation principle is composed of two sub-principles: (*i*) the *entity/element preservation* principle [65], and (*ii*) the *axiom preservation* principle, also called the *entailment deduction satisfaction* [66]. The knowledge preservation principle applies to cases of *simple-merge* and *full-merge* ontology integration. However, in the case of *asymmetric merge*, it applies only on the *target*/priority ontology. Still, the knowledge of the *source*/non-priority ontologies should be preserved as much as possible in the integrated ontology.

5.1.3. Alignment Preservation Principle

The principle of *alignment preservation*, *a.k.a. correspondences preservation* [64] or *equality preservation* [65], stands for the preservation of all correspondences coming from the alignments that are used in the ontology integration. The preserved correspondences should not only include *equivalence* correspondences (≡), but should also include *subsumption/is-a* correspondences (⊑/⊒) if the used alignment contains such correspondences. Thus, the ontology integration process should preserve all types of relations contained in the used alignment(s), such as *equivalence*, *subsumption*, *disjointness*, *etc*.

5.1.4. Ontology Coherence Principle

Before defining the *coherence* principle, let us clarify some ambiguous related notions. The three following terms are used in the literature in a confusing manner: *(un)satisfiability*, *(in)consistency* and *(in)coherence*.

Satisfiability/Unsatisfiability [67]. This notion is related to entities (classes and properties). An unsatisfiable entity is an unrealizable entity containing a false or contradictory description, which means that no instance can meet all the requirements to be a member of that entity. Recall that a class is instantiated by individuals; and a property is instantiated by pairs of individuals (and data literals): an individual in the property domain, and an individual or a data literal in the property range. An unsatisfiable class can (and should) never have instances, like does the class *owl:Nothing*, because there will be no instance that can satisfy it [67]. Similarly, an unsatisfiable property can (and should) never have instances, because there will be no instance that can satisfy it, like do the properties *owl:bottomObjectProperty* and *owl:bottomDataProperty*.

Consistency/Inconsistency [68]. This notion is related to ontologies. An ontology is consistent if it has a satisfying interpretation. For example, an ontology from which we deduce that an individual x is different from and identical to an individual y, cannot have a satisfying interpretation. Inconsistency can occur whenever there is at least one violation of a class restriction, one instantiation of an unsatisfiable class or property, one instantiation of two disjoint classes or properties, or one semantic contradiction between individuals in the A-box like in the previous example, *etc*. Since an inconsistent ontology has no model, all of its classes and properties become unsatisfiable, *i.e.* none of its classes and properties can be instantiated. It is considered as a severely damaged ontology, containing a serious error that must be repaired because no useful knowledge can be inferred from it by ontology reasoning. In addition, it cannot be published nor deployed in applications that require a logical reasoning process [68].

Coherence/Incoherence [67]. This notion refers to ontologies. An ontology is called *coherent* when all of its classes and properties are satisfiable. If there is at least one unsatisfiable class (except *owl:Nothing*) or one unsatisfiable property (except *owl:bottomObjectProperty* and *owl:bottomDataProperty*) in an ontology, then the latter becomes incoherent. An incoherent ontology is still consistent, thus it can be published and used in applications. However, if one of its unsatisfiable entities is instantiated, then it would become inconsistent. Therefore, in an incoherent ontology, an inconsistency may always occur [67].

Coherence $\Rightarrow$ Consistency	Incoherence $\not\Rightarrow$ Inconsistency
Coherence $\not\Leftarrow$ Consistency	Incoherence $\Leftarrow$ Inconsistency

In the literature, the *coherence principle* is called the *consistency principle*. However, we will rather use the *coherence* term which seems to us more appropriate. The *coherence* principle states that all entities of the integrated ontology should be satisfiable, assuming that the input ontologies also do not contain any unsatisfiable entities [44]. In other words, it states that there is no additional unsatisfiable class or property in the integrated ontology. The coherence principle actually aims at the coherence (and the consistency) of the integrated ontology, assuming that input ontologies are coherent too.

If logical reasoning is involved in an application, then ensuring coherence becomes mandatory because the integrated ontology must be logically/semantically correct to support reasoning and be really useful. Indeed, in this case, an incoherent ontology may lead to incomplete or unexpected results, although it can be used in other applications. For example, in a text annotation application, it is not necessary to ensure coherence in the integrated ontology because the annotation task does not involve a reasoning process. However, in other applications such as query answering, logical errors may have a critical impact on the query answering process.

Violation of the Coherence Principle

An integrated ontology may contain a large number of unsatisfiable information. Whenever we have to reason on the integrated ontology (*i.e.*, on the input ontologies and their initially computed correspondences), it is often necessary to detect unexpected semantic conflicts or contradictions. This is caused by two main reasons or their combination [55]: (*i*) correspondences, generated especially by an automatic matching tool, may be incorrect and contain erroneous correspondences; and (*ii*) even if all the correspondences have been found to be correct, the input ontologies may have heterogeneous conceptualizations, organizations or structuring of the same entities: That is, they may have contradictory descriptions of the matched entities. Besides, the alignment correspondences are not independent of each other [32]: In some cases, only one among several correspondences can be true (*See* Example 1); In other cases, several correspondences, put together, may lead to a false conclusion mainly caused by *disjointness* axioms of the input ontologies (*See* Example 2).

Example 1 (Coherence Violation). *Let us assume that we have a class A in O_1, two disjoint classes B and C in O_2, and two correspondences c_1 and c_2 stating that A is a subclass of B and C. Formally,*

$$O_1 = \{A\} \quad O_2 = \{B \perp C\}$$
$$\mathcal{A} = \{c_1, c_2\} \quad c_1 = < A \sqsubseteq B > \quad c_2 = < A \sqsubseteq C >$$

If a reasoning process is applied to the integrated ontology O_3, then A will be an unsatisfiable class since it will become a subclass of two disjoint classes.

Example 2 (Coherence Violation). *Consider the example proposed by Fahad et al. [69]. Suppose that in O_1, we have two disjoint classes, "Employee" and "Student", and a class "PhD_Researcher" which is a subclass of "Student". While in O_2, "Employee" and "Student" are not disjoint, and "PhD_Researcher" is a subclass of both "Employee" and "Student". Formally,*

$$O_1 = \{Employee \perp Student,\ PhD\_Researcher \sqsubseteq Student\}$$
$$O_2 = \{Employee,\ Student,\ PhD\_Researcher \sqsubseteq Employee,$$
$$PhD\_Researcher \sqsubseteq Student\}$$

If a reasoning process is applied to the ontology O_3, then "PhD_Researcher" will be an unsatisfiable class since it will become a subclass of two disjoint classes.

Errors such as unsatisfiable classes and unsatisfiable properties (*i.e.*, ontology incoherence) and ontology inconsistency are very likely to occur in an integrated ontology. They are reflections of the unintended logical inferences that are still difficult to detect beforehand, understand, explain, and repair. As a consequence, an integrated ontology will always be prone to errors, which will lower its performance and make it non-understandable and even unusable.

5.1.5. Ontology Conservativity Principle

The principle of *conservativity* requires that the integrated ontology does not introduce new semantic relations between entities of each (or at least one) of the input ontologies, especially new *subsumption* relations causing structural changes [44, 70, 71]. In other words, the original description of an input ontology, especially the *is-a* structure of its class hierarchy, should not be altered after being integrated. Consequently, the behavior of the services already functioning with that input ontology is not affected by the use of the new integrated ontology.

Violation of the Conservativity Principle

The integrated ontology may contain a large number of conservativity violations, which are not limited only to classification and structural changes (if we are dealing with the unrestricted conservativity problem). Like coherence violations, conservativity violations reveal either erroneous correspondences or an incompatibility/disagreement between the input ontologies [44]. However, they may also evidence incompleteness in (one of) the input ontologies [44].

Example 3 (Conservativity Violation). *Consider the following two ontologies: O_1 contains two classes A and B; and O_2 contains two classes A' and B' where B' is a subclass of A'. Formally,*

$$O_1 = \{A, B\} \quad O_2 = \{B' \sqsubseteq A'\}$$
$$\mathcal{A} = \{c_1, c_2\} \quad c_1 = < A \equiv A' > \quad c_2 = < B \equiv B' >$$

Suppose that we have to integrate O_2 into O_1, that is to say that the description of O_1 should not be altered after the integration process. If the ontology matching step generates two correspondences c_1 and c_2 stating that A is

equivalent to A′, and B is equivalent to B′, then the original structure of O_1 will change because of the addition of a new subsumption relationship linking A and B after a full merge.

In the following subsections, we shed light on general ontology requirements dealing with *redundancy of entities*, *subsumption cycles*, *redundancy of the structure*, *redundancy of relations*, *multiplicity of property domains and ranges*, and *unconnected entities*, *etc*. Although they are not problematic from a semantic/logical point of view, these issues do undoubtedly reduce the conciseness of the integrated ontology and should be better avoided in any ontology (not only in integrated ones). We can find general ontology requirements in many ontology evaluation works, *e.g*, [65, 66, 72, 73].

5.1.6. Minimality Principle

The principle of *minimality* [74] states that no redundant entities should appear in an integrated ontology, assuming that the input ontologies (that have been integrated) also do not contain any of them. This principle means the *prohibition of entity redundancy* or *duplication*. Entities are called *redundant* or *duplicated* whenever they do have the same meaning, and therefore, whenever they represent the same entity in an ontology.

Redundancy & Ambiguity of Entities

Redundancy or *duplication* of entities in an (integrated) ontology occurs whenever there are distinct but actually equivalent entities coming from different input ontologies. They can also have a large overlap in their terms, which makes them *ambiguous* entities, *i.e.* having a high degree of ambiguity. *Ambiguity* is assessed through the number of labels appearing in multiple entities of a given ontology [42]. These common entities are redundant because they are neither linked by *equivalence* relations, nor merged with each other in the integrated ontology. Ontology integration should not introduce additional duplication and ambiguity. Indeed, redundant entities will increase the size of the integrated ontology, complicate text annotation tasks due to ambiguity, and decrease the interoperability between applications that use these entities [42]. Indeed, a service of an application that is using a class X cannot interoperate with a service of another application that is using an equivalent class Y, since an *equivalence* axiom linking these two classes is missing [42]. However, in an integrated ontology, ensuring the coherence and conservativity principles will lead to numerous redundant entities.

5.1.7. Acyclicity Principle

The principle of *acyclicity* [65, 66] or *cycle prohibition* states that no *subsumption* cycles should appear in the class or property hierarchy of an integrated ontology, assuming that the input ontologies also do not contain any of them. Subsumption cycles have the form $\{e_1 \sqsubseteq e_2, \ldots, e_n \sqsubseteq e_1\}$, where $n > 1$ and e are entities of the same type (classes or properties) [42]. Subsumption/*is-a* relations should be acyclic because they complicate graph-based algorithms such as path extraction, hierarchy traversal and depth counting, leading to an undetermined depth (hierarchy level) in the class or property hierarchy [42]. Consequently, ontology integration should not introduce additional cycles. However, unexpected cycles may usually appear in an integrated ontology. Figure 6 shows an illustrative example of a *subsumption* cycle that we will explain later in section 8.

5.1.8. Principle of Structural and Relational Redundancy Prohibition

We propose another principle called *prohibition of structural and relational redundancy*. The latter states that no structural or relational redundancies should appear in an integrated ontology, assuming that the input ontologies also do not contain any of them.

Structural Redundancy

Structural Redundancy or briefly *redundancy* [75], *a.k.a. semantic overlap* [27] or *semantic redundancy* [27], is the redundancy that occurs in the class hierarchy, *i.e.*, whenever there is more than one *is-a* path from a class to a parent other than the root [75]. In other words, it happens whenever there is more than one *is-a* path from the ontology root to a leaf concept. These paths are called *leaf paths* [27]. This type of redundancy is caused by the multiple inheritance resulting from a full merge. We think that this requirement is too strict and can be relaxed, because the multiple inheritance exists in many real-world ontologies and does not necessarily lead to semantic/logical conflicts. Similarly, *instance overlap* [63, 64] happens whenever an individual is instantiated by more than one concept/class.

Example 4 (Structure Redundancy). *Consider two ontologies O_1 and O_2. In O_1, "PhD_Student" is a subclass of "Student". In O_2, "PhD_Student" is a subclass of Researcher.*

$$O_1 = \{PhD\_Student \sqsubseteq Student\} \quad O_2 = \{PhD\_Student \sqsubseteq Researcher\}$$
$$O_3 = \{PhD\_Student \sqsubseteq Student,\ PhD\_Student \sqsubseteq Researcher\}$$

In the fully merged ontology O_3, "PhD_Student" will be a subclass of two classes, each coming from a different ontology. However, a given class can be subsumed by a single super-class that encompasses the union of two or more classes. So, "PhD_Student" can also be subsumed by a class that is the union of "Student" and "Researcher":

$$O_3 = \{PhD\_Student \sqsubseteq (Student \cup Researcher)\}$$

Relational Redundancy

Redundant information in an ontology is any explicit information that can already be inferred by an ontology reasoner. Redundant relations are implicitly mentioned in the ontology and useless to repeat [76]. In short, redundant relations or edges are those that are exactly repeated or those that can be deduced by other paths (*i.e.*, by another sequence of relations). Unexpected relation redundancies may appear in an integrated ontology, especially the redundancy of *subsumption* relations. This is caused by the full merge of entities or by the addition of *equivalence* relations linking different entities in an integrated ontology. *Equivalence* relations are actually shortcuts for pairs of *subsumption* relations. Indeed, an *equivalence* relation between two entities (*i.e.*, an *equivalentClass* or *equivalentProperty* axiom) is implicitly equal to two *subsumption* relations (*i.e.*, two *subClassOf* or *subPropertyOf* axioms) in both directions, as stated in equation 1 where e_1 and e_2 are two entities (*i.e.*, two concepts/classes or two object/datatype properties).

$$\langle e_1, e_2, \equiv \rangle = \langle e_1, e_2, \sqsubseteq \rangle + \langle e_2, e_1, \sqsubseteq \rangle \tag{1}$$

The redundancy of relations occurs when a merged concept subsumes or is subsumed by a *source* concept and a *target* concept that are linked by a *subsumption* or *equivalence* relation [77]. It also occurs when a merged individual is instantiated by a *source* concept and a *target* concept that are linked by a *subsumption* or *equivalence* relation [77]. There are many other redundancy cases as in the examples below:

Example 5 (Relation Redundancy). *Consider the following two ontologies O_1 and O_2. In O_1, A is a parent class of B. While in O_2, C is a parent class of D. Formally,*

$$O_1 = \{B \sqsubseteq A\} \quad O_2 = \{D \sqsubseteq C\}$$
$$\mathcal{A} = \{c_1, c_2\} \quad c_1 = < A \sqsubseteq D > \quad c_2 = < B \sqsubseteq C >$$

If the ontology matching step generates two correspondences c_1 and c_2 stating that A is a subclass of D, and B is a subclass of C, then the integrated ontology O_3 will yield a redundancy in the subsumption relations. The subsumption between B and C is redundant because it is already expressed by the three consecutive subsumptions between B and A; A and D; and D and C.

Example 6 (Relation Redundancy). *Consider another example also inspired by [69]. In O_1, we have two disjoint classes, "Employee" and "Student". In O_2, we have not only "Employee" and "Student" classes which are not disjoint, but also "PhD_Researcher" and "Non_PhD_Researcher" classes which are sub-classes of "Student", such that "Employee" is disjoint with "Non_PhD_Researcher". Formally,*

$$O_1 = \{Employee \perp Student\}$$
$$O_2 = \{PhD\_Researcher \sqsubseteq Student,\ Non\_PhD\_Researcher \sqsubseteq Student,$$
$$Employee \perp Non\_PhD\_Researcher\}$$

In the fully merged ontology O_3, the disjointness between "Non_PhD_Researcher" and "Employee" (originating from O_2) is already expressed by the disjointness between "Employee" and "Student" (originating from O_1).

5.1.9. Property's Domain and Range Oneness

The principle of *property's domain and range oneness* [66, 78] (or the *prohibition of property's domain and range multiplicity*) states that object and datatype properties should have only one domain and only one range in the integrated ontology. When it comes to the range of datatype properties, this principle is specifically referred to as the principle of *one datatype restriction* [66]. The latter states that datatype properties should have only one range (*i.e.*, only one datatype) in the integrated ontology. Usually, when we assign multiple domains/ranges to a property, we implicitly want to say that the property has a union of domains/ranges (*i.e.*, has a domain/range that is composed of a disjunction of

classes (or datatypes)). However, multiplicity has a totally opposite meaning when interpreted in a logical way. Indeed, although the OWL language allows a property to have multiple domains and ranges, it interprets this multiplicity as an intersection of domains or ranges (*i.e.*, as a conjunction of classes or datatypes). Therefore, the ontology integration process should not introduce properties having multiple domains and/or ranges. Properties should rather have only one domain/range that composes the multiple domains/ranges using the union expression <*owl:unionOf*> [24].

It should be noted that domain and range statements are not constraints to be checked [79]. They are inference axioms (*i.e.*, used by the ontology reasoner to infer further knowledge). In fact, domain and range statements are global restrictions (*i.e.*, that have a global scope [80]). They do not behave as property restrictions that just restrict the property when it is associated with a particular class; they do restrict the property globally. For example, if we have a property *P* having a domain *D* and a range *R* (*i.e.*, <*P*, *rdfs:domain*, *D*> and <*P*, *rdfs:range*, *R*>), individuals *a* and *b*, and a property assertion <*a*, *P*, *b*>, then the reasoner will infer that *a* is an instance of *D* (<*a*, *rdf:type*, *D*>) and that *b* is an instance of *R* (<*b rdf:type*, *R*>) [81]. A domain or range violation happens when *a* (or *b*) is not an instance of *D* (or *R*). However, violating a domain or range constraint does not necessarily lead to an ontology inconsistency, but it can cause unexpected inferences that can lead to errors [79]. An ontology inconsistency will only be generated by a reasoner if the domain/range is constituted by multiple classes (or datatypes) that are disjoint with each other [81]. In this case, no individual (or no data literal) can be an instance of these disjoint domains or ranges.

Example 7 (Domain and Range Multiplicity). *Consider the two ontologies O_1 and O_2. Both ontologies have the object property "studiesIn". In O_1, "studiesIn" has the class "Person" as a domain and the class "School" as a range. In O_2, "studiesIn" has the class "Student" as a domain and the class "University" as a range. Formally,*

$$O_1 = \{Person \xleftarrow{domain} studiesIn \xrightarrow{range} School\}$$
$$O_2 = \{Student \xleftarrow{domain} studiesIn \xrightarrow{range} University\}$$
$$O_3 = \{Person, Student \xleftarrow{domain} studiesIn \xrightarrow{range} School, University\}$$
$$O_3 = \{(Person \cap Student) \xleftarrow{domain} studiesIn \xrightarrow{range} (School \cap University)\}$$

In the fully merged ontology O_3, the property "studiesIn" will be merged and will have the two domains "Person" and "Student", and the two ranges "School" and "University". An ontology reasoning process will interpret the domain of "studiesIn" as the intersection (conjunction) of the two classes "Person" and "Student", and the range of "studiesIn" as the intersection (conjunction) of the two classes "School" and "University". The following object property assertion declares that "Ines" (which is an instance of the class "Student") studies in "Tunis_El_Manar_University" (which is an instance of the class "University"):

$$O_3 = \{< Ines \longrightarrow studiesIn \longrightarrow Tunis\_El\_Manar\_University >\}$$

If a reasoning process is performed on the integrated ontology O_3, the reasoner will infer that "Ines" is an instance of "Person" and an instance of "Student". Similarly, the reasoner will infer that "Tunis_El_Manar_University" is an instance of "School" and an instance of "University". If the class "Person" was disjoint with the class "Student" or if the class "School" was disjoint with the class "University", the ontology O_3 would be inconsistent, because "Ines" and "Tunis_El_Manar_University" would be instances of two disjoint classes, which is impossible. When the classes "Person" and "Student" are not disjoint and the classes "School" and "University" are not disjoint (which is our case), the reasoner will not find any inconsistency (unless it finds one in another part of the ontology due to these inferences). A correct integration process would assign to the property "studiesIn" a domain composed of the union (disjunction) of "Person" an "Student", and a range composed of the union (disjunction) of "School" and "University", as follows:

$$O_3 = \{(Person \cup Student) \xleftarrow{domain} studiesIn \xrightarrow{range} (School \cup University)\}$$

Example 8 (Datatype Restriction Multiplicity). *Consider an example proposed by Babalou and König-Ries [66]. Suppose that we have two ontologies O_1 and O_2. Both ontologies have the datatype property "author_Id". In O_1, "author_Id" has the class "Person" as a domain and the datatype "String" as a range. In O_2, "author_Id" has the class "Person" as a domain and the datatype "Integer" as a range. Formally,*

$$O_1 = \{Person \xleftarrow{domain} author\_Id \xrightarrow{range} String\}$$
$$O_2 = \{Person \xleftarrow{domain} author\_Id \xrightarrow{range} Integer\}$$
$$O_3 = \{Person \xleftarrow{domain} author\_Id \xrightarrow{range} String, Integer\}$$

$$O_3 = \{Person \xleftarrow{domain} author\_Id \xrightarrow{range} (String \cap Integer)\}$$

In the fully merged ontology O_3, the property "author_Id" will be merged, and will have the domain "Person" and the two ranges "String" and "Integer". An ontology reasoning process will interpret the range of "author_Id" as the intersection (conjunction) of the two datatypes "String" and "Integer". The following datatype property assertion declares that "Ines" (which is an instance of the class "Student") has the author ID "author_1234" (which is a literal of the datatype "String"):

$$O_3 = \{< Ines \longrightarrow author\_Id \longrightarrow \text{“}author\_1234\text{”} >\}$$

If a reasoning process is performed on the integrated ontology O_3, the reasoner will infer that "Ines" is an instance of "Person". Similarly, the reasoner will infer that "author_1234" is a value of the datatype "String" and a value of the datatype "Integer". This will lead to the inconsistency of O_3 (which is a fatal error), because the literal "author_1234" is not a String and an Integer at the same time. If the range datatypes had an overlap (such as the datatypes "Int", "Integer", "unsignedInt", "nonNegativeInt", and "positiveInt"; or the datatypes "Literal" and "PlainLiteral", etc), the ontology O_3 could avoid being inconsistent. A correct integration process would assign to the property "author_Id" a range composed of the union (disjunction) of "String" and "Integer", as follows:

$$O_3 = \{Person \xleftarrow{domain} author\_Id \xrightarrow{range} (String \cup Integer)\}$$

5.1.10. Entity Connectivity

The principle of *entity connectivity* [66, 78] or *unconnected entity prohibition* states that no entity (class or property) should be isolated or unconnected in a given ontology. In other words, it states that entities that originally have some connections in the input ontologies should preserve these connections and not become totally unconnected in the integrated ontology. An unconnected entity is an entity that has no relation to the rest of the ontology. More precisely, a class or a property becomes unconnected when it does not have any associated *subsumption* relations in its description [78], thus it does not have any named parents, children or siblings. This definition can be relaxed by considering an entity unconnected when it also does not have any associated *non-taxonomic* relations in its description [78], such as *equivalence* or *disjointness* relations for classes and properties, and *inverseOf* relations for object properties. The ontology integration process can make some entities unconnected, especially the asymmetric integration case. The latter allows to remove some relations from the *source* (non-priority) ontologies in order fulfill coherence and/or conservativity principles for the *target* (priority) ontology.

5.2. Ontology Integration Repair Strategies

To ensure the integration principles, we have noticed two repair/debugging perceptions widely used in state-of-the-art ontology integration research works. Some authors choose to alter the alignments because they are considered to be *incoherent* and *non-conservative* with respect to the input ontologies, while the input ontologies are considered to be always sound and much more reliable than alignments. Other authors choose to alter the input ontologies because they consider that whenever the alignments contain all the correct correspondences and yet the integrated ontology still contains *coherence* and *conservativity* violations, then these violations are only caused by the incompatible models of the input ontologies which have conceptual differences of the same domain. Finally, others choose to alter both the input ontologies and the alignments because they consider that conflicts can be caused either by alignments or by ontologies. Actually, the target of the repair approach is case dependent. For example, in Multi-Agent Systems, each agent uses a private knowledge encoded as an OWL ontology that cannot be altered, which makes the only possible repair solution is to agree on a common alignment between these agents. It may be the case that one or all (or none) of the input ontologies should be immutable.

State-of-the-art approaches generally remove elements from the alignments and/or from the input ontologies. They may also rarely add elements to the input ontologies for the correction of the *conservativity* principle. All these strategies should satisfy the principle of *minimal change*, sometimes called *minimality*. Its purpose is to avoid removing or adding axioms as much as possible, based on the number of removed correspondences from the alignments, or the number of removed axioms from the input ontologies, or rarely based on the number of inserted axioms to the input ontologies.

5.2.1. Alignment Improvement

We distinguish two types of alignment improvement [17]: the *alignment repair* which can be performed in the ontology matching module or in a separate module; and the *alignment disambiguation* which can also be used in the alignment repair.

Alignment Repair. Alignment *repair* or *repairing*, (*a.k.a.* alignment *debugging*, alignment *diagnosis*, *mapping revision*, *mapping repair*, or briefly *repair*), aims at resolving the violations of *coherence* and/or *conservativity* by reducing the alignments and preserving the input ontologies being integrated.

An alignment should comply with the *coherence* principle. A *coherent alignment* [51, 82] is an alignment that does not violate the *coherence principle* by not introducing unsatisfiable entities in the integrated ontology. During the ontology matching step, matching tools should exploit the hidden semantics of the input ontologies, especially the *disjointness* information, to generate *coherent* correspondences. Current alignment repair tools are mostly based on the satisfiability checking, such as *ALCOMO* [51, 61], *LogMap* [83, 84], *AML* [43, 85], *YAM++* [86], *ASMOV* [87], and *Lily* [88] repair facilities. They compute justifications by searching for the set of axioms entailing each unsatisfiable entity in the integrated ontology, then remove the minimal set of correspondences involved in the justifications. This process is called the *minimal diagnosis process* [89]. On the one hand, *ALCOMO* makes a complete reasoning for the justification by computing all possible justifications for each unsatisfiable class. Justifications can be *disjointness* axioms, *domain restriction* axioms, *range restriction* axioms, and all types of axioms in $\mathcal{EL}^{++}$ [90, 91] which is a fragment and a lightweight version of DL. This makes *ALCOMO* a non-scalable alignment repair tool because using the complete reasoning in the repair process is very expensive and inefficient. However, *ALCOMO* lets the user choose between complete or incomplete reasoning. On the other hand, *LogMap*-Repair and *AML*-repair make an incomplete reasoning that mainly computes one justification for each unsatisfiable class, namely a *disjointness*-based justification. They eliminate the incoherent correspondences that are involved in conflicts caused by *disjointness* relations between classes of the input ontologies. To address the scalability problem, occurring whenever there is a large number of unsatisfiabilities, most of these approaches rely on approximate algorithms, and thus perform an *approximate(d) alignment repair*. Apart from filtering out the incoherence-causing correspondences, some approaches, such as *LogMap*, may also alter the alignment by changing some *equivalence* correspondences to *subsumption* correspondences—like does the second method of the *alignment disambiguation* technique in the next paragraph.

An alignment should also comply with the *conservativity* principle. A *conservative alignment* is an alignment that does not violate the *conservativity* principle. It does not introduce new semantic relations between entities of an input ontology, especially new *subsumption* relations, in the integrated ontology. Correspondences leading to such alterations should be discarded from the alignment. In general, state-of-the-art approaches such as [55, 70, 71] works capture the amount of structural change between two given ontologies using the notion of *logical difference* [92, 93] or *deductive difference* [55]. In particular, they often use the *approximation of the deductive difference* [44] that captures the amount of structural change between the classification hierarchies of two given ontologies. These techniques compare the *subsumption* entailments of the resulting integrated ontology with the *subsumption* entailments of an input ontology—before being integrated. The purpose is to detect the set of *subsumption* entailments (*i.e.*, inferred axioms) that do hold in the integrated ontology but do not hold in the original input ontology. This difference reflects the new *subsumption* entailments that were introduced to the input ontology after being integrated.

The alignment repair will restore the *coherence* and/or the *conservativity* of the produced alignment, but will generate redundancy/duplication and ambiguity in the integrated ontology.

Alignment Disambiguation. It aims at disambiguating or reducing a *many-to-many* alignment to obtain a *one-to-one* alignment. It corresponds to a *bipartite matching* problem. There are two approaches for dealing with ambiguous *equivalence* correspondences. The choice of the appropriate approach is case dependent [42].

1. The first approach considers that among the ambiguous *equivalence* correspondences, there is only one correct correspondence that truly reflects a synonym or alternative entity, while the remaining ones rather reflect similar, related or overlapping terms that do not strictly denote equivalent entities [42]. The simplest method to resolve the disambiguation problem is to always keep the *equivalence* correspondence that has the highest confidence value, and remove the remaining ones [17]. This approach consists in filtering correspondences having the same source entity or the same target entity by only keeping a single correspondence having the highest confidence value (*See* Figure 8). It is a greedy approach [17] that can be viewed as a *Stable Marriage/Matching* problem, which assigns one object from a first set to one and only one object from a second set [94].
   An alternative solution is based on the *principle of locality*, which assumes that entities semantically related to entities of a correct correspondence are likely to be matched to each other by correct correspondences too. Semantically related entities are hierarchy neighbors such as more general entities (*i.e.*, parents/ancestors) and more specific entities (*i.e.*, children/descendants). Low confidence in the neighborhood of a correspondence can

reveal its incorrectness. For example, if entities e_1 in O_1 and e_2 in O_2 are correctly matched, then the neighbors of e_1 are likely to be matched to the neighbors of e_2. If the correspondences relating the neighbors of e_1 and e_2 have low confidence values, then the correspondence $< e_1 \equiv e_2 >$ may be incorrect. Therefore, for each ambiguous correspondence, this disambiguation method counts the confidence proportion of correspondences reachable by super-entities and/or sub-entities of the matched entities, and finally only keeps the correspondence that has the highest ratio [95].

2. If one of the two matched ontologies is more granular or general than the other one (meaning that the terms of one ontology are more granular or general than those of the other ontology), then the ambiguous *equivalence* correspondences are considered as *subsumption* correspondences [96]. Indeed, an entity in the general ontology is decomposed into several entities in the granular ontology. Therefore, the second method to tackle the disambiguation problem proposes to refine *equivalence* relations to *subsumption* relations through changing their type from "$\equiv$" to "$\sqsubseteq$" or "$\sqsupseteq$" [42, 44, 96].

5.2.2. Ontology Improvement

We distinguish three types of ontology improvement: the *ontology repair* and the *ontology correction*, which are performed after the ontology matching module; and the *ontology refinement*, which is performed after the ontology merging module.

Ontology Repair *(Ontology Axiom Exclusion).* Ontology *repair*, *a.k.a.* ontology *diagnosis*, ontology *debugging*, or ontology *revision*, aims at fixing the violations of *coherence* and/or *conservativity* by reducing the input ontologies being integrated and preserving the produced alignment between them. Once a violation is detected, the strategy of axioms exclusion can be applied to solve it. It removes a minimal set of axioms (from the input ontologies) involved in logical conflicts and/or alterations in the integrated ontology, as do *Babylon Health* [42] and *ContentMap* [55]. Nevertheless, doing so leads to the loss of ontological knowledge that could sometimes be critical or of paramount importance for some applications or services.

Ontology Correction *(Ontology Axiom Inclusion).* Ontology correction aims at fixing the violations of *conservativity* by increasing the input ontologies being integrated and preserving the produced alignment between them. Once a *conservativity* violation is detected, the strategy of axioms inclusion can be applied to solve it. It adds to the input ontologies (each in isolation) a minimal set of axioms, such that they can entail novel axioms that were originally violating the *conservativity* principle in the integrated ontology [44, 97, 98].

Ontology Refinement or Pruning. Although structural redundancies, relational redundancies, *subsumption* cycles and unconnected entities do not semantically or logically affect the coherence of the integrated ontology, it is often necessary to refine or prune the integrated ontology to keep it understandable, sound and simple [77]. Refining an integrated ontology prunes its relational and structural redundancies, resolves its *subsumption* cycles, and removes (or connects) its unconnected entities, *etc*. To solve the cycles, Pottinger and Bernstein [65] came up with the idea of merging the cycle into a single entity and keeping all the properties of its combined entities. The intuition behind this is that the *is-a* relation is transitive; hence, a cycle of *is-a* relations implies the equality of all its elements. As for the *target*-driven asymmetric merge, Raunich and Rahm [63, 64] proposed to remove from the cycle one of the *is-a* relations that are originating from the *source* ontologies. Indeed, according to them, a full merge of two acyclic ontologies cannot involve only *source* relations or only *target* relations in a cycle. Similarly, to solve structural redundancies introduced in their integrated ontology, they proposed to remove the additional leaf paths that are originating from the *source* ontologies. Finally, to solve the unconnected concepts in their integrated ontology, Raunich and Rahm [63, 64] removed the *source* concepts that are no longer linked to any concept (via a *subsumption* relation).

Remark 1. Some authors (*e.g.*, in [77, 78, 99]) consider the ontology repair (or ontology debugging) as one of the ontology refinement tasks. However, in this paper, we wanted to highlight two different ontology integration repair approaches, which are the *alignment* repair and the *ontology* repair. Therefore, we preferred to make a separation between ontology *repair* and ontology *refinement*. Indeed, we consider that *ontology repair* mainly resolves semantic/logical issues (such as ontology inconsistency and incoherence) which are serious errors in an ontology, whereas *ontology refinement* resolves additional issues that are much less serious because they do not involve semantics.

We conclude that whenever ontologies describe conflicting domain perspectives, then blindly integrating them and ensuring both *coherence* (& *conservativity*) and *information preservation* from ontologies and alignments is infeasible.

6. Ontology Integration: Related Work

In this section, we review the process of ontology integration, summarize the different integration methods used in the case of multiple input ontologies, then outline the most important research works on ontology integration, and finally put forth some initial observations.

6.1. General Ontology Integration Process

The general workflow of the ontology integration process covers the following main phases [100]:

1. **Pre-processing Phase**: It analyzes, evaluates and validates the selection of the input ontologies, normalizes them, and/or improves their quality, mainly whenever they originally contain conflicts or redundancies, in order to reduce the matching phase workload. For instance, if the ontologies to be matched are not expressed in the same language [101], then a normalization step is necessary to translate them from one language or formalism into a uniform representation (*e.g.*, OWL) without changing their semantics;
2. **Matching Phase**: It identifies correspondences between the input ontologies—usually pairs of equivalent entities—and generates an alignment. It includes the following steps [32]: filtering/hashing, entity similarity comparison—using syntactic, structural and/or semantic matchers—, correspondences generation, and alignment repair (optional). A *repairing* step can be either performed during the matching step, or performed separately in a standalone step;
3. **Merging Phase**: It merges the selected input ontologies into an integrated ontology (*See* Subsection 4.2);
4. **Post-processing Phase**: It evaluates, repairs, and refines the resulting ontology by checking its consistency and coherence, resolving its cycles and its coherence & conservativity violations, and pruning its redundancies.

6.2. Integration of Multiple Ontologies

Most of the state-of-the-art research works have addressed the issue of integrating or merging only two input ontologies. In the case of more than two input ontologies, ontology *integration* or *enrichment* is carried out always in an incremental way, while ontology *merging* can be performed incrementally or non-incrementally. In the following, we introduce the notions of *pairwise* or *binary ontology integration* and *holistic* or *N-ary ontology integration* to respectively denote the incremental and non-incremental methods. Both approaches produce the same resulting integrated ontology.

6.2.1. Pairwise Ontology Integration

In the incremental ontology integration approach, an empty or initial *target* ontology O^* is iteratively fed and extended with a set of *source* (or *local*) ontologies $O_1, O_2, \ldots, O_n$, as shown in Figure 2. For each iteration, the available pairwise ontology alignment $\mathcal{A}$ involving the two current ontologies being processed is used (*e.g.*, in [102, 103]). When the number of input ontologies is equal to N, the integration process of these ontologies makes N iterations (if we count the initialization step). In the example of Figure 2, the integration process goes through five iterations for integrating five ontologies:

1. $O^* = O_1$, then
2. $O^* = O_{12} = O_1 + O_2 + \mathcal{A}_{1-2}$, then
3. $O^* = O_{123} = O_{12} + O_3 + \mathcal{A}_{12-3}$, then
4. $O^* = O_{1234} = O_{123} + O_4 + \mathcal{A}_{123-4}$, then
5. $O^* = O_{12345} = O_{1234} + O_5 + \mathcal{A}_{1234-5}$.

During the ontology integration process, O^* is called an *intermediate* integrated ontology [78] and it gets larger after each iteration. At the end of the process, O^* is called a final *target* (or *global*) integrated ontology. The post-processing (or refinement) step for O^* can be performed progressively after each iteration, or at the end of the integration process. Normally, the order in which the *source* ontologies are included in the *target* ontology does not influence the resulting integrated *target* ontology. In the schema integration domain in general, the pairwise integration approach is called the *ladder strategy* [104]. This approach is not scalable because it becomes impractical when there is a large number of input ontologies.

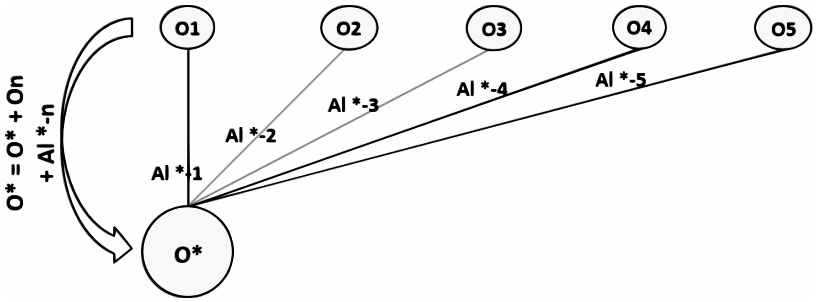


Figure 2: Pairwise Ontology Integration

6.2.2. Holistic Ontology Integration

In the non-incremental ontology integration approach, all the *source* (or *local*) ontologies O_1, O_2, ..., O_n are combined together to constitute the *target* (or *global*) integrated ontology O^*. In this case, the whole process is carried out through a single iteration using the available holistic ontology alignment $\mathcal{A}$ involving all the *source* ontologies, as shown in Figure 3. In the example of Figure 3, the ontology integration process makes one iteration:

- $O^* = O_{12345} = O_1 + O_2 + O_3 + O_4 + O_5 + \left[\mathcal{A}_{12345}\right]$

Otherwise, the whole process can be carried out in a single iteration using pairwise alignments between all ontology pairs, like in a normalized network of ontologies, as shown in Figure 4 (*e.g.*, in [78, 99, 105, 106]). This method can be considered as the integration of a network of ontologies (*See* Subsection 3.6). In the schema integration domain in general, this method is called the *balanced binary strategy* [104]. In the example shown in Figure 4, the ontology integration process performs one iteration:

- $O^* = O_{12345} = O_1 + O_2 + O_3 + O_4 + O_5 + \left[\mathcal{A}_{1-2} + \mathcal{A}_{1-3} + \mathcal{A}_{1-4} + \mathcal{A}_{1-5} + \mathcal{A}_{2-3} + \mathcal{A}_{2-4} + \mathcal{A}_{2-5} + \mathcal{A}_{3-4} + \mathcal{A}_{3-5} + \mathcal{A}_{4-5}\right]$

The holistic ontology integration approach is scalable because it is suitable for a large number of input ontologies.

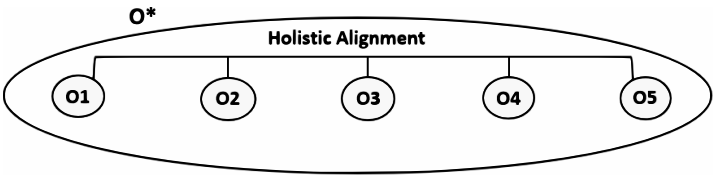


Figure 3: Holistic Ontology Integration

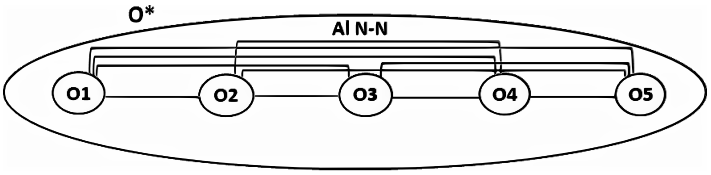


Figure 4: Holistic Ontology Integration using Pairwise Alignments

The *pairwise ontology integration* is much more adopted in the literature than the *holistic ontology integration*, where the latter remains a challenging task to perform. This is due to two reasons:

- First, the ontology matching community has always adopted the pairwise matching, because of its simple search space, and has rarely considered the holistic matching, which explains the unavailability of holistic ontology alignments;

- Second, the current automated ontology matching systems have become quite proficient at generating pairwise ontology alignments, and particularly at identifying simple *equivalence* correspondences between two ontologies [32].

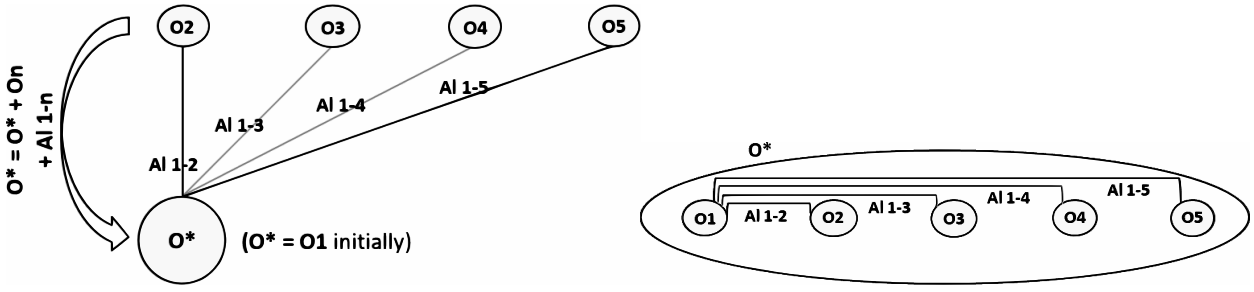
Remark 2. In some works, the integration is performed using only the pairwise alignments between the *target* ontology in its initial state and the *source* ontologies. They proceed in an incremental manner as shown in Figure 5a (*e.g.* in [42]) or in a non-incremental manner as shown in Figure 5b (*e.g.* in [107]). These two processes do not really differ. In the example of Figure 5a, the integration process makes five iterations:

1. $O^* = O_1$, then
2. $O^* = O_{12} = O_1 + O_2 + \mathcal{A}_{1-2}$, then
3. $O^* = O_{123} = O_{12} + O_3 + \mathcal{A}_{1-3}$, then
4. $O^* = O_{1234} = O_{123} + O_4 + \mathcal{A}_{1-4}$, then
5. $O^* = O_{12345} = O_{1234} + O_5 + \mathcal{A}_{1-5}$.

Similarly, in Figure 5b, the ontology integration process makes one iteration:

- $O^* = O_{12345} = O_1 + O_2 + O_3 + O_4 + O_5 + \left[\mathcal{A}_{1-2} + \mathcal{A}_{1-3} + \mathcal{A}_{1-4} + \mathcal{A}_{1-5}\right]$

Their downside is that they do not achieve a complete semantic interoperability between the ontologies that have been integrated. Indeed, *source* ontologies (O_2, O_3, O_4, O_5) may share common entities that do not exist in the *target* ontology (O_1). These common entities between the *source* ontologies will be redundant because they are not matched with each other. Thus, they are neither linked nor merged in the integrated ontology (O^*).



(a) Incremental/Pairwise Method (b) Non-Incremental/Holistic Method

Figure 5: Ontology Integration Resulting in an Incomplete Semantic Interoperability

6.3. Scrutiny of the Common Approaches

In this subsection, we review the most relevant state-of-the-art research works related to the ontology integration. Research works that are only limited to the ontology matching or the ontology repair are beyond the scope of our paper.

Commonly, the ontology integration techniques can be assessed through the following criteria: the number of input ontologies to be integrated (*i.e.*, two or more ontologies), the level of user involvement (*i.e.*, automation or semi-automation), the information loss or incompleteness (*i.e.*, preservation of the alignments and ontologies knowledge), the quality of the integrated ontology (*i.e.*, violations, redundancies, cycles, *etc*), and the scalability. Table 3 sums up the most prominent approaches and their limitations. The mentioned works are listed in a chronological order. If a tool is semi-automatic or manual, then it cannot scale up, although achieving better results than do the automatic tools. Similarly, (coherence & conservativity) *repair* and (ontologies & alignments) *information preservation* can never be simultaneously achieved.

Table 3: Summary of Related Works

Tool / Approach	#Input Ontologies	Tasks	Strategy	User Involvement[1]	Ontologies' Preservation[2]	Alignment Preservation[2]	Coherence Repair[2]	Conservativity Repair[2]	Scalability[1]
PROMPT (2000) [75, 108]	2	matching & merging	full merge	semi-automatic	×	×	×	×	×
Chimaera (2000) [109]	2	matching & merging	full merge	semi-automatic	×	×	×	×	×
FCA-Merge (2001) [110]	2	matching & merging	full merge	semi-automatic	×	×	×	×	×
ONION (2001) [103]	2 or + (pairwise)	matching & merging	simple merge	semi-automatic	✓	✓	×	×	×
OntoMerge (2003) [111]	2	merging	simple merge	automatic	✓	✓	×	×	—
HCONE (2006) [112]	2	matching & merging	full merge	semi-automatic	×	×	×	×	×
SAMBO (2006) [7]	2	matching & merging	full merge	semi-automatic	×	×	×	×	×
ILIADS (2007) [113]	2	matching & merging	simple merge	automatic	✓	×	✓	×	—
ContentMap (2009) [55]	2	merging	simple merge	semi-automatic	×	×	✓	✓	×
— (2010) [114, 115]	2	matching & merging	simple merge	automatic	✓	✓	×	×	×
ATOM (2011) [63, 64]	2	merging	asymmetric merge	automatic	×	×	×	×	✓
DKP-AOM (2012) [116]	2	matching & merging	full merge	automatic	×	×	✓	×	—
FITON (2014) [117]	2 or + (holistically)	matching & merging	simple merge	semi-automatic	×	×	×	×	✓
— (2015) [102]	2 or + (pairwise)	matching & merging	simple merge	manual	✓	✓	×	×	×
— (2016) [107]	2 or + (holistically)	matching & merging	full merge	semi-automatic	×	×	×	×	×
— (2017) [106]	2 or + (holistically)	matching & merging	full merge	automatic	×	×	×	×	×
OIM-SM (2017) [26]	2	matching & merging	full merge	automatic	×	×	×	×	×
Babylon Health (2018) [42]	2 or + (pairwise)	matching & merging	asymmetric merge	automatic	×	×	✓	✓	×
BMOnto (2019) [49]	2	matching & merging	asymmetric merge	automatic	✓	✓	×	×	✓
CoMerger (2020) [78, 99]	2 or + (holistically)	matching & merging	full merge	automatic	✓	✓	×	×	✓

[1] *Semi-automation* (*i.e.*, manual intervention) and *Scalability* can never be simultaneously fulfilled.

[2] *Knowledge Preservation* and *Violations Management* can never be both fulfilled at the same time.

6.4. Observations on the Related Work

By taking a closer look at the related work, we make the following observations:

1. To avoid dealing with *coherence* violations in the integrated ontology, we notice that many integration and merging approaches do only preserve the hierarchy/taxonomy of the input ontologies (mainly the early research works) or do not particularly preserve the *disjointness* knowledge of the input ontologies (mainly the recent research works), *e.g.*, [26, 27, 63, 64, 75, 106, 109], *etc*. Indeed, ontologies follow the Open World Assumption (OWA). The latter states that the concepts of an ontology are supposed to be shared and reused by multiple applications and users, because they are meant for an open distributed world such as the Web. Therefore, concepts are not disjoint by default—they do overlap—and their description should be "closed off" where appropriate by *disjointness* relations. That is why, there are no unsatisfiable entities without *disjointness* axioms. It should be noted that coherence violations may also be caused by the use of disjointness correspondences, or by the use of implicit disjointness axioms not directly stated in the input ontologies. However, disjointness knowledge of the input ontologies can be of paramount importance in some applications, and removing it can sometimes be critical.
2. It is worth mentioning that many ontology integration works (mainly the early ones) do not evaluate their integrated ontology. They only focus on evaluating the quality of their ontology alignments resulting from the ontology matching step, *e.g.* in [7, 26, 106, 107, 112–117].
3. We do also observe that most of the tools, in particular the early ones, are usually semi-automatic, requiring a lot of human intervention to either validate the suggested correspondences in the matching step, or to validate the actions to be performed in the merging step. This is especially the case when it comes to *subsumption* correspondences. Human intervention is also needed to resolve violations and redundancies either during or after the construction of the integrated ontology. Therefore, the result is usually dependent on the user observation or the expert decisions, *e.g.*, in *PROMPT* [75, 108], *Chimaera* [109] and *SAMBO* [7], *etc*.
4. Finally, many research works are not generic. They are tailored to integrate only two ontologies that cover a predefined specific domain, since the integration of more than two ontologies at the same time is more complex.

By and large, the way in which the issue of ontology integration is handled has gradually evolved and matured over time. In the last decade, research works generally focused on providing a graphical interface for an interactive real-time visualization in the process of ontology matching and merging. In doing so, the user had the upper hand to accept or reject the results and visually compare entities of the input ontologies, *e.g.*, in *PROMPT* [75, 108], *Chimaera* [109], *SAMBO* [7] and *iMERGE* [118]. They only preserve the class hierarchy of the input ontologies and therefore only treat structural redundancies in the integrated ontology, *e.g.*, in [26, 63, 64, 75, 109]. They also emphasize on the matching process by only evaluating the correctness and completeness of the alignment, and ignoring the evaluation of the integrated ontology, *e.g.*, in [7, 26, 106, 107, 112–117]. In addition, they are generally able to integrate only two ontologies. However, nowadays, recent ontology integration approaches tend to be more generic, making it possible to jointly integrate more than two ontologies. Besides, they are moving towards full automation and scalability of their approaches. Unlike the previous works, recent ones try to preserve as much information as possible from both alignments and ontologies, and therefore treat all kinds of violations, redundancies, and cycles in the integrated ontology. They also focus on assessing the quality of the alignment as well as the integrated ontology.

7. Ontology Integration: Evaluation Metrics

Looking at the related work approaches from the previous section, we notice that there are various input ontologies, different ontology integration types, many input parameters, and diverse evaluation metrics being used in the studied papers. There is a clear tendency towards the use of *Precision* and *Recall* metrics in the early works for assessing the matching results (*i.e.*, the alignments). However, recent works use other evaluation metrics for assessing the integrated ontology, *e.g.* number of unsatisfiable classes, logical difference, number of cycles, number of redundant entities, *etc*. This makes it hard to compare these ontology integration approaches and make general statements on their performance.

Evaluating ontology integration techniques is still an open issue. It is difficult to make a comparison between an integration result and a gold standard because there are no agreed quality metrics/measures for evaluating them, such as *Precision*, *Recall*, and *F-Measure* metrics in the Information Retrieval field. We suggest using *Precision* and *Recall*

measures to compute false and missing axioms of a resulting integrated ontology compared to an ideal integrated ontology. However, there is a huge lack of references within the community. Besides, a perfect result is impossible to manually obtain for large ontologies, and there could be more than just one perfect result [27]. For the time being, there are no accepted benchmarks and gold standard criteria that can be used to objectively and generally assess the quality of the proposed integration approaches. There have been some attempts to produce benchmarks for ontology integration. The first existing benchmark [27] is not published and thus cannot be used, whereas the second one [119] is composed of only very small ontologies. Both proposed benchmarks result from an automatic asymmetric merge of two input ontologies. Inspired by [27, 42, 55, 64, 69, 78, 119] respective works, we gather the following metrics that can assess ontology integration results:

1. *Entities coverage*: Number of preserved entities from the input ontologies;
2. *Axioms coverage/completeness*: Number of preserved input ontologies' axioms;
3. *Correspondences coverage*: Number of preserved alignment correspondences;
4. *Ontology consistency*: Is the integrated ontology consistent? (True or False);
5. *Ontology coherence*: Number of unsatisfiable entities in the integrated ontology;
6. *Logical difference* of the input ontologies before and after being integrated;
7. *Redundancy*: Number of redundant/duplicated entities in the integrated ontology;
8. *Ambiguity*: Number of overlapping terms appearing in the entities of the ontology;
9. *Leaf path difference* of the input ontologies before and after being integrated;
10. *Loops*: Number of *subsumption* cycles in the integrated ontology;
11. Number of *properties with multiple domains or ranges* in the integrated ontology;
12. Number of *unconnected entities* (classes & properties) in the integrated ontology;
13. *Efficiency*: The runtime performance;
14. *Scalability*: Scalable runtimes when using large and rich input ontologies;
15. *Manual effort*: Is the user involved in the ontology integration process?

The three first metrics are meant to evaluate the degree of *information preservation* or *completeness*. Coverage metrics for entities and axioms satisfy the principle of *ontologies' knowledge preservation* (*See* Subsection 5.1.2), while coverage of correspondences satisfies the principle of *alignment preservation* (*See* Subsection 5.1.3). They ensure that there is no information loss or incompleteness from the input ontologies and alignments. The metric of *entities coverage* [27, 64] can be split into *classes coverage*, *object properties' coverage*, *data properties' coverage*, and *instances coverage*. It reflects (the number or) the percentage of preserved entities in the resulting integrated ontology compared to the expected number of entities:

- For the *simple merge* case, the number of entities of the integrated ontology should be ideally equal to the sum of the entities of the input ontologies.
- For the *full merge* of two ontologies, the number of entities of the integrated ontology should be ideally equal to the sum of entities of the input ontologies, minus the number of merged entities (*i.e.*, minus the number of *equivalence* (≡) correspondences of the used *non-ambiguous pairwise* alignment).
- For the *asymmetric merge* case, the number of entities of the integrated ontology should be greater than the number of entities of the initial *target* ontology, and generally less than the sum of all entities of the input *target* and *source* ontologies.

The metric of *axioms coverage* [42] reflects the number (or the percentage) of preserved axioms in the integrated ontology. In addition to this metric, we can add two more specific metrics, that are *subsumption axioms coverage* and *disjointness axioms coverage*, to make sure that these particular types of axioms are preserved, since they are prone to be removed during the integration process. In the case of *simple-merge* and *full-merge* ontology integration, the metrics of entities and axioms coverage apply to all the input ontologies being integrated. That is, entities and axioms of the input ontologies should ideally be completely preserved. However, in the case of *asymmetric-merge* ontology integration, these *(overall)* metrics can be split into *source* coverage and *target* coverage [27]. Consequently, we can

have six metrics: On the one hand, *source entities coverage*, *target entities coverage*, and *(overall) entities coverage*; On the other hand, *source axioms coverage*, *target axioms coverage*, and *(overall) axioms coverage*. The entities and axioms of the *target* ontology must be completely preserved. However, some entities and axioms of the *source* ontologies can be missed. Both metrics of *source* entities and axioms coverage can be useful when comparing the *source* coverage of different ontology integration tools, in order to show the tool that most preserves the knowledge of the *source* ontologies. The metric of *correspondences coverage* [64] reflects the number (or the percentage) of preserved correspondences in the integrated ontology. When the used alignment also contains *subsumption* correspondences, the metric of correspondences coverage can be split into *equivalence correspondences coverage* (=), *"is-a" correspondences coverage* (<), and *"inverse is-a" correspondences coverage* (>) [64].

Both of the fourth and fifth metrics satisfy the *coherence* principle (*See* Subsection 5.1.4). They assess the logical consistency of the integrated ontology and find the number of its unsatisfiable entities, mainly its unsatisfiable classes. The *logical difference* metric, denoted by $\texttt{diff}^{\approx}$ [44, 55] or |LDiff| [42], fulfills the *conservativity* principle (*See* Subsection 5.1.5). It finds, in the integrated ontology, the number of axioms that have been added to the original description of the input ontologies. In other words, it is the difference in the number of axioms of the input ontologies before and after being integrated. The two metrics of *entities redundancy* and *entities ambiguity* fulfill the *minimality* principle in the integrated ontology (*See* Subsection 5.1.6). They can be divided into *classes redundancy* and *classes ambiguity*, *properties redundancy* and *properties ambiguity*, and *instances redundancy* and *instances ambiguity* [66].

The number of *loops* (or *is-a* cycles) fulfills the *acyclicity* principle in the integrated ontology (*See* Subsection 5.1.7). We can divide this metric into *class cycles* and *property cycles* [66]. The metric of *leaf path difference*, denoted by Δ `leaf paths` [63, 64], satisfies the principle of *structural redundancy prohibition* in the integrated ontology (*See* Subsection 5.1.8). It is the difference in the number of leaf *is-a*-paths of the input ontologies before and after being integrated. This number reflects the amount of structural redundancies introduced after the integration. As for the principle of *relational redundancy prohibition* (*See* Subsection 5.1.8), to the best of our knowledge, there is no existing metric that can reflect the redundancy of relations or paths in an ontology. The number of (object and datatype) *properties that have multiple domains or ranges* satisfies the principle of *property's domain and range oneness* in the integrated ontology (*See* Subsection 5.1.9). The number of *unconnected classes and properties* in the integrated ontology satisfies the principle of *entity connectivity* (See Subsection 5.1.10). Metrics 5, 6, 7, 8, 9, 10, 11, and 12 should ideally be equal to 0, but they do strongly depend on the quality of the input ontologies that have been integrated.

Efficiency and *scalability* are performance evaluation criteria. *Efficiency* means that the runtime of the ontology integration algorithm should compete with runtimes of the existing algorithms. As for the *scalability* criterion, the algorithm should be able to provide good performance and acceptable runtimes for large and heavyweight ontologies having hundreds of thousands of entities and axioms. Finally, *manual effort* is a user-related evaluation criterion. The user or expert intervention should be minimal, and it would be even better if the algorithm was fully automatic without any human intervention.

Although they belong to the ontology matching domain, *Precision*, *Recall*, and *F-Measure* can be included for evaluating the quality of the used alignment against a *reference* alignment. These three metrics are adopted by the Ontology Alignment Evaluation Initiative (OAEI)[6] campaign. They ensure the *alignment correctness and completeness* principle (*See* Subsection 5.1.1).

8. Application Case for Future Investigation

In this section, we focus on the integration of a network of ontologies, *i.e.* where we perform a holistic ontology integration using pairwise alignments (*See* Subsection 6.2.2). We aim to investigate the challenges of this particular ontology integration case, which illustrates the issues being faced, and then elaborate a set of learned lessons for the topic from the results of this experimentation.

8.1. General Process of the Experimentation

Nowadays, there are many available good-quality alignments resulting from well-known ontology matching systems. Leveraging them will help have trustworthy ontology integration results. We propose to holistically integrate two

[6] http://oaei.ontologymatching.org/

or more independently developed ontologies, using *reference* alignments between all ontology pairs, as shown in Figure 4. To do so, we have developed two algorithms: OIAR (Ontology Integration with Alignment Reuse) and AROM (Alignments Reuse for Ontology Merging) to automatically build a *simple-merge* ontology and a *full-merge* ontology, respectively. Both of them have been developed using the OWL API [120], which is a Java API for developing, manipulating, and serializing OWL ontologies. OIAR and AROM take as input two or more OWL input ontologies to be integrated, one or more RDF alignments described in the Alignment API format [40], a new IRI or URI as a namespace for the future integrated ontology, and a confidence threshold ranging between [0, 1] to trim correspondences of the input alignment(s), *i.e.* to exclude correspondences below a given confidence/similarity value. They preserve all entities and axioms from the input ontologies and all correspondences from the input alignments (*See* Table 4). Source code and results of OIAR[7] and AROM[8] can be downloaded from the GitHub platform.

Table 4: Comparison between OIAR and AROM

Tool	#Input Ontologies	Tasks	Strategy	User Involvement	Ontologies' Preservation	Alignment Preservation	Coherence Repair	Conservativity Repair	Scalability
OIAR	2 or + (holistically)	merging	simple merge	automatic	✓	✓	×	×	✓
AROM	2 or + (holistically)	merging	full merge	automatic	✓	✓	×	×	✓

We have chosen to carry out the experiments on the *Large Biomedical Ontologies* (*LargeBio*) track provided by the OAEI 2020 campaign. Ontologies, reference alignments and participant alignments are all downloadable from the OAEI website. *LargeBio* is composed of three large and semantically rich ontologies (*See* Table 5), namely *FMA* (Foundational Model of Anatomy), *SNOMED-CT* (Clinical Terms), and *NCI* (National Cancer Institute Thesaurus).

Table 5: Number of Entities in the *LargeBio* Ontologies

LargeBio	**Classes**	**Object Prop.**	**Data Prop.**	**Instances**	**Logical Axioms**
FMA	78,988	0	54	0	79,218
NCI	66,724	123	67	0	96,046
SNOMED-CT	122,464	55	0	0	191,203
Total	268,176	178	121	0	366,467

Table 6: Number of Correspondences in the *LargeBio* Reference Alignments

Alignment	**Original**			**Disambiguated**		
	≡	?*	**Total**	≡	?*	**Total**
FMA-NCI	2,686	338	3,024	2,369	190	2,559
FMA-SNOMED	6,026	2,982	9,008	5,209	2,579	7,788
SNOMED-NCI	17,210	1,634	18,844	13,606	790	14,396
Total Correspondences	25,922††	4,954	30,876†	21,184‡‡	3,559	24,743‡

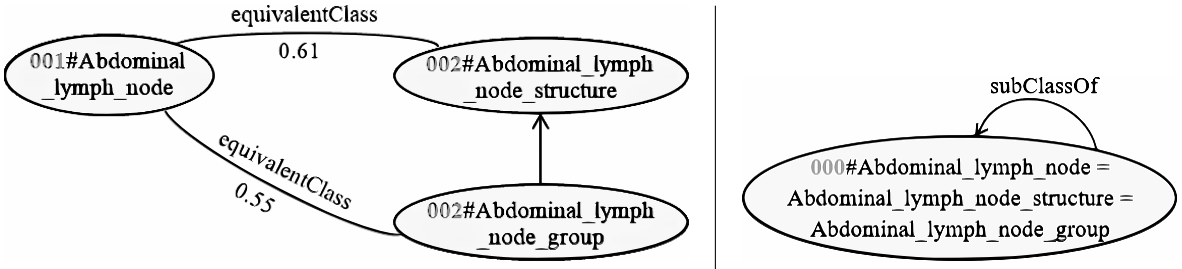
* When these incoherence-causing correspondences are deleted, the alignment becomes repaired.
† The *original* reference alignments contain 30,876 correspondences.
†† The *repaired* reference alignments contain 25,922 correspondences.
‡ The *disambiguated* reference alignments contain 24,743 correspondences.
‡‡ The *disambiguated & repaired* reference alignments contain 21,184 correspondences.

The OAEI benchmark provides reference alignments between each pair of the *LargeBio* ontologies based on the UMLS meta-thesaurus [121] (*See* Table 6). In the reference alignments, correspondences having relations flagged by

[7] https://github.com/inesosman/OIAR
[8] https://github.com/inesosman/AROM

"?" are correct *equivalence* correspondences involved in coherence violations detected by *ALCOMO*, *LogMap*, and/or *AML* alignment repair facilities. As a result, *LargeBio* reference alignments are only repaired for coherence violations, not also for conservativity violations.

We have integrated the three *LargeBio* ontologies using their three pairwise reference alignments—between each pair of them. The IRI of our integrated output ontology is *http://integration*. In the remainder, we omit IRI/URI prefixes of the entities for readability reasons. All tests have been performed with a confidence threshold equal to 0.0, thus we have kept all the correspondences of the input alignments. It should be recalled that the higher the input confidence threshold, the lower the number of unsatisfiable entities in the integrated ontology. In addition, integrating ontologies from different domains always generates fewer unsatisfiable classes than integrating ontologies from the same domain. In the following, we explain the reasons for adding an alignment disambiguation module in some of our tests.



(a) in a Simple-Merge Ontology (b) in a Full-Merge Ontology

Figure 6: Two Ambiguous *Equivalence* Correspondences Leading to Redundancies and Cycles

Example 9. *Figure 6 illustrates a snippet from the integration of FMA (O_1) and SNOMED-CT (O_2). It shows subsumption redundancies and a subsumption cycle formed because of the addition of two equivalence correspondences having the same source class "001#Abdominal_lymph_node". In fact, in the simple-merge ontology (Figure 6a), each equivalence axiom linking two classes is formally equivalent to two subsumption axioms in both directions. However, in the full-merge ontology (Figure 6b), the three equivalent classes are merged together to constitute a single class that becomes subsumed by itself, since the subsumption between "002#Abdominal_lymph_node_group" and "002#Abdominal_lymph_node_structure" classes is conserved. This conserved subsumption is redundant because any class is subsumed by itself by default.*

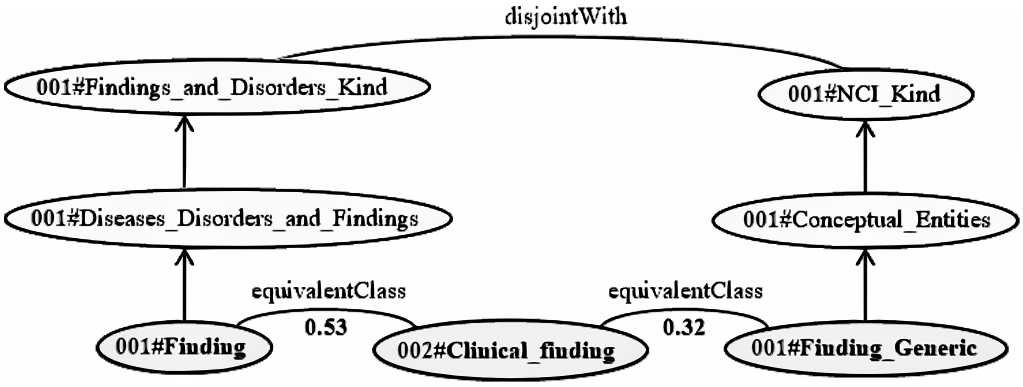
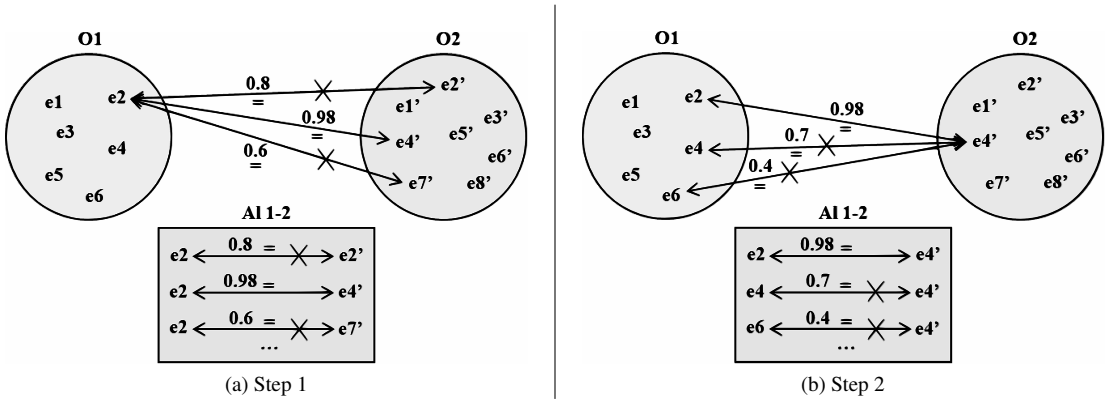


Figure 7: Two Ambiguous *Equivalence* Correspondences Leading to Unsatisfiable Classes in a Simple-Merge Ontology (or Leading to an Unsatisfiable Class in a Full-Merge Ontology)

Example 10. *Figure 7 depicts a snippet from the integration of NCI (O_1) and SNOMED-CT (O_2). It represents unsatisfiable classes in dark grey, formed by the addition of two ambiguous equivalence correspondences having the*

same source class "002#Clinical_finding". The three classes "001#Finding", "002#Clinical_finding" and "001#Finding_Generic" become, by inference, sub-classes of the two disjoint classes "001#Findings_and_Disorders_Kind" and "001#NCI_Kind" where the disjointness information comes from NCI (O_1). This contradiction makes the three classes unsatisfiable.

In this case, if the DisjointWith axiom is removed, then we will avoid all these unsatisfiable classes. Nevertheless, the structure/hierarchy of the first input ontology (NCI) will be altered, and thus the conservativity principle will be violated. Indeed, the class "001#Finding_Generic" will be a subclass of the class "001#Finding" and all its parents, and the class "001#Finding" will be a subclass of the class "001#Finding_Generic" and all its parents, which was not originally stated in this ontology at first. However, if the class "002#Clinical_finding", that belongs to the second input ontology (SNOMED-CT), is matched to only one class from the other ontology, precisely to the class "001#Finding" with which it has the highest similarity value, then we will avoid all these unsatisfiable classes, conserve the disjointness axiom, and satisfy the conservativity principle. Therefore, the alignment repair process would remove the correspondence between "002#Clinical_finding" and "001#Finding_Generic".



(a) Step 1 (b) Step 2

Figure 8: Ontology Alignment Disambiguation (Approach 1)

For this reason, in some tests, we disambiguate the ambiguous *equivalence* correspondences. The disambiguation step is explained in subsection 5.2.1 and depicted in Figure 8. In this disambiguation, we filter out correspondences having the same *source* entity (as shown in Figure 8a), then we filter out correspondences having the same *target* entity (as shown in Figure 8b) by only keeping the most confident correspondence that has the highest value. If it happens that all ambiguous correspondences have exactly the same confidence value, we have decided to keep all of them, because if we randomly choose one of them, then the results will differ for each chosen correspondence.

8.2. Experimentation Results

Tables 7 and 8 sketch the quality of the ontologies resulting from integrating *LargeBio* ontologies using OIAR and AROM respectively, such that all axioms of the input ontologies are preserved. The *Original* column means that we have kept the input alignments ambiguous and unrepaired (*i.e.*, we have used all existing correspondences from the input alignments). The *Disambiguated* column means that we have disambiguated the input alignments by only keeping one correspondence from each set of ambiguous correspondences (*i.e.*, we have used all correspondences from the disambiguated input alignments). The *Repaired* column means that we have repaired the input alignments by removing their correspondences that have a relation "?" (*i.e.*, we have only used correspondences having relations "≡" from the input alignments). The *Disambiguated & Repaired* column means that we have disambiguated and repaired the input alignments (*i.e.*, we have only used correspondences having relations "≡" from the disambiguated input alignments).

Example 11. *After the integration of FMA (O_1), NCI (O_2) and SNOMED-CT (O_3), this is an example of the unsatisfiable classes that were introduced in our resulting integrated ontology despite the repair and the disambiguation of the input reference alignments. Figure 9 shows two unsatisfiable classes: "001#Portion_of_cytosol" from FMA and "002#Cytoplasmic_Matrix" from NCI. After adding two non-ambiguous equivalence correspondences, they become, by*

inference, sub-classes of the two disjoint classes "001#Portion_of_body_substance" and "001#Anatomical_structure", where the disjointness information comes from FMA. Here, to ensure the coherence of the integrated ontology, we will face a dilemma between sacrificing an equivalence correspondence from the alignment, which will reduce interoperability between ontologies, or sacrificing the disjointness axiom from the input ontology FMA, which will be a knowledge loss.

It is important to mention that when we integrate the three *LargeBio* ontologies using all correspondences from the original (unrepaired and ambiguous) reference alignments and without conserving any *DisjointWith* axiom from the input ontologies, we do not get any unsatisfiable class in our integrated ontology. In this case, our integrated ontology is consistent and coherent, but incomplete, *i.e.*, lacking valuable disjoint knowledge. This proves that *disjointness* axioms are the main cause of semantic conflicts in the integration of *LargeBio* ontologies. It should also be noted that the *full merge* ontology always generates fewer unsatisfiable entities than does the *simple merge* ontology because it naturally contains fewer entities after they have been fully merged. Nevertheless, performing a full merge or a simple merge is exactly the same from a semantic point of view. In other terms, if one leads to unsatisfiable entities or an inconsistency, then the other will do so; and if one does not lead to unsatisfiable entities or an inconsistency, then the other will do so.

8.3. Takeaway Lessons from the Experimentation

In the following, we draw the attention of the reader to two lessons that we can figure out from our experimentation results:

Lesson 1: When we get all these unsatisfiable classes in the integrated ontology, we can doubt the correctness of the OAEI reference correspondences. OAEI reference alignments are directly extracted from the UMLS metathesaurus, which is the most comprehensive effort for integrating independently developed medical terminologies and ontologies. Pesquita *et al.* [123] proved that the original unrepaired reference alignments of the OAEI *LargeBio* track do contain erroneous correspondences [124], and that *ALCOMO*, *LogMap* and *AML* repair facilities, used by *LargeBio* reference alignments, do sometimes remove or alter correct correspondences. Overall, *LargeBio* reference alignments do contain some erroneous or missing correspondences, but even if they were totally perfect, we still cannot escape unsatisfiabilities. In fact, in a network of ontologies, incoherence can come either from a *local incoherence* in a particular ontology or alignment, or from a *global incoherence* between them [105]. However, in the case here, we believe that these unsatisfiable classes are beyond the abilities of the common alignment repair systems. It is worth noting that the current alignment repair systems do not deal with the simultaneous integration of multiple ontologies, and are dedicated to only integrating two ontologies using an alignment between them. Indeed, if we integrate two *LargeBio* ontologies using a repaired reference alignment between them, then we obtain a consistent and coherent ontology, *i.e.* that has no unsatisfiable classes. However, if we integrate more than two ontologies using several repaired pairwise alignments (between ontology pairs), we obtain an ontology that can have considerable unsatisfiable classes. This proves the compelling need for alignment repair systems that would be able to deal with networks of ontologies. In [105], Euzenat pointed out new perspectives on this challenging issue worth of exploration, namely, repairing networks of ontologies.

Lesson 2: Let us compare the two columns *Disambiguated* alignments and *Repaired* alignments from Tables 7 and 8. We observe that although the alignment disambiguation removes more correspondences than does the alignment repair, the use of the disambiguated alignments generates much more unsatisfiable classes than the use of the repaired alignments. We deduce that the alignment disambiguation is an "aggressive" approach that removes unnecessary correspondences without being able to guarantee coherence in the integrated ontology. It should be noted that the repaired alignments do contain ambiguous correspondences (as shown in Table 6). Despite being ambiguous, *LargeBio* repaired reference alignments do not lead to any unsatisfiable classes when we use them to integrate each *LargeBio* ontology pair separately. In other words, when we integrate a *LargeBio* ontology pair using its repaired reference alignment, we do not get any unsatisfiable classes in the integrated ontology. In this case, disambiguating the repaired alignment is useless and will generate many more redundant classes in the integrated ontology. To sum up, it is important to know that not all ambiguous *equivalence* correspondences generate unsatisfiable entities in the integrated ontology. However, alignment repair approaches may include an alignment disambiguation step in some cases, whenever needed.

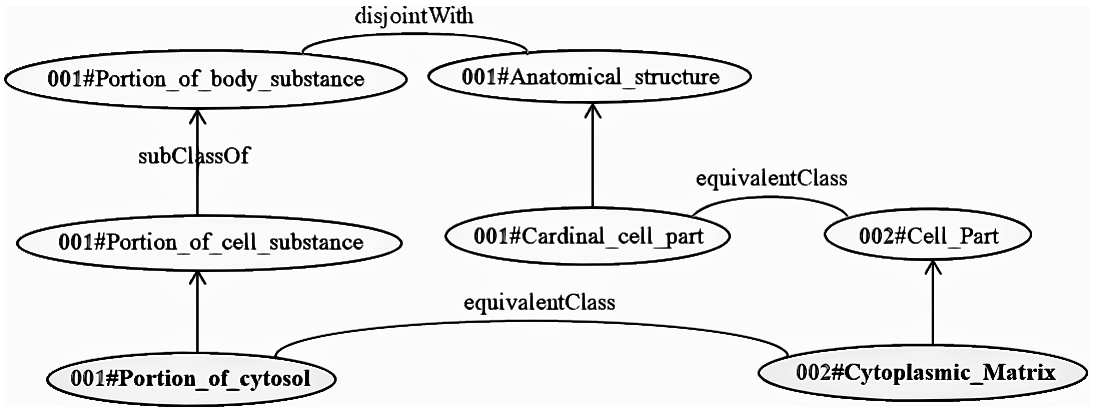


Figure 9: Unsatisfiable Class Formation in a Simple-Merge Ontology

Table 7: Quality of the Ontology Resulting from the Integration of *LargeBio* Ontologies using OIAR

Simple-Merge Integrated Ontology Features	**Input Alignments**			
	Original	Repaired	Disambiguated	Disambiguated & Repaired
Classes			268,176	
Object Properties			178	
Datatype Properties			121	
Instances			0	
Logical Axioms[1]	397,343 (366,467 + 30,876)	392,389 (366,467 + 25,922)	391,210 (366,467 + 24,743)	387,651 (366,467 + 21,184)
Consistency[2]	✓	✓	✓	✓
Unsatisfiable Classes[2]	203,675	49,046	155,775	43,078
Redundant Classes	0	8,498	4,501	10,540
Runtime[3] (min)	0.70	0.68	0.71	0.70

[1] In parentheses, 366,467 is the total number of axioms of all input ontologies, and the value after the sum operator is the total number of correspondences of all input alignments.
[2] Computed using ELK, which is one of the fastest $\mathcal{EL}$ reasoners for large ontologies [122].
[3] Runtimes do not include matching times since we take pre-established alignments as input.

Table 8: Quality of the Ontology Resulting from the Integration of *LargeBio* Ontologies using AROM

Full-Merge Integrated Ontology Features	**Input Alignments**			
	Original	Repaired	Disambiguated	Disambiguated & Repaired
Classes	240,634	244,173	245,334	248,097
Object Properties			178	
Datatype Properties			121	
Instances			0	
Logical Axioms	359,600	360,577	362,404	363,135
Consistency[1]	✓	✓	✓	✓
Unsatisfiable Classes[1]	177,975	42,450	138,523	38,067
Redundant Classes	0	8,498	4,501	10,540
Runtime[2] (min)	0.76	0.75	0.72	0.77

[1] Computed using the ELK ontology reasoner [122].
[2] Runtimes do not include matching times and alignment repair times.

9. Findings from the Survey

In this section, we derive some interesting findings and future directions in the ontology integration area with respect to issues related to the alignment and ontology repair, ontology refinement, ontology matching sub-areas, and the evaluation of the integrated ontology.

9.1. Ontology Integration Repair

If the integration goal is interoperating different applications, then we believe that we should be strict with respect to the preservation of the input ontologies. Thus, no axiom from the input ontologies is allowed to be removed, and we should rather sacrifice correspondences by only repairing alignments, although this will decrease the semantic interoperability of applications. However, if the integration goal is to create a new ontology for a particular purpose, then we believe that the process can be relaxed with respect to the preservation of the input ontologies. Thus, removing axioms from the input ontologies can be tolerated, and the fulfillment of the conservativity principle is no longer mandatory. Consequently, we can preserve all correspondences from the original alignment in order to avoid the redundancy and ambiguity of entities in the resulting integrated ontology.

Since simple- and full- merge will always be torn between *completeness*, *coherence* (and also *conservativity*) such that we can never obtain a perfect result, the asymmetric merge or enrichment turns out to be the most successful integration type. At least, it surely achieves *completeness* and *conservativity* for the target *seed* ontology, and it surely achieves *coherence* for the resulting target integrated ontology. We think that it is particularly well suited to the integration case that aims to build an application-specific ontology. To sum up, integrating ontologies with incompatible/conflicting domain views or models is still an open and thriving issue [123]. However, if the asymmetric merge (enrichment) is acceptable in some ontology integration scenarios, then it can be a possible solution to this issue.

When dealing with violations, we can observe that most state-of-the-art repair approaches often choose the *alignment* repair over the *ontology* repair. Ontology integration works prefer repairing ontology alignments because there are already many ontology matching tools that perform with alignment repair, contrary to ontology repair that is much less experimented in the ontology integration area. However, many ontology integration scenarios actually suit to a more balanced repair approach between the input alignments and the input ontologies. The most common limitations in alignment repair works are scalability and complete reasoning. Besides, if there are no *disjointness* axioms declared in the input ontologies, then this may badly affect the quality of the alignment repair results.

In the reported ontology integration literature, we can underscore that the *conservativity* principle is much less investigated than the *coherence* principle. Indeed, tools that automatically detect and repair correspondences leading to a logical incoherence in the integrated ontology are the most widely proposed. However, tools that automatically detect and repair correspondences leading to changes in the original description of the input ontologies are still to be investigated and improved. Moreover, current conservativity repair systems only deal with the structural changes introduced in the integrated ontology. In addition, they generally only focus on repairing the conservativity of a single input ontology, which means that they can only resolve the asymmetric merge or enrichment case where the only repair target is the *seed* (priority) ontology. Overall, there is hardly any approach—among the examined ontology integration approaches—that deals with both coherence and conservativity violations at the same time, except the works of Stoilos *et al.* [42] and Jiménez-Ruiz *et al.* [55].

9.2. Ontology Refinement

All the research works that are mentioned in this survey did not consider refining the resulting ontology after the ontology integration process, except the work of Babalou and König-Ries [78, 99]. Therefore, another finding from this survey is that ontology refinement tools are largely missing in the ontology integration literature and still need to be addressed (*See* Subsection 5.2.2).

9.3. Ontology Matching

First, as mentioned before, it is known that most state-of-the-art ontology integration approaches are generally limited to a simple pairwise ontology matching module. In fact, throughout the last two decades, there has been a large number of approaches dealing with pairwise ontology matching. They have increased not only in number but also in performance. This is due to the availability of many benchmarks for evaluating simple pairwise ontology matching approaches.

Second, in open, distributed and heterogeneous systems such as the Semantic Web, matching and integrating networks of ontologies (*i.e.*, the holistic ontology matching and integration) may be very beneficial to further improve interoperability and scalability. However, most of the current ontology integration approaches are limited to matching and integrating only a pair of ontologies, while only a few systems manage multiple ontologies at once. This is due to the pairwise ontology matching strategy that gained much more attention than did the holistic ontology matching one. Holistic ontology matching is rarely considered in the literature because of its complexity and the lack of benchmarks for evaluating holistic matching approaches. As a result, current alignment repair systems also do not deal with the integration of networks of ontologies. As discussed in Section 8, they are still dedicated to only repairing an alignment between two ontologies. Future works on the ontology alignment repair area can be oriented towards this direction.

Third, most research works in the area of ontology matching remain focused on the identification of simple *equivalence* correspondences between ontological entities, which is the simplest ontology matching case. The performance of this task is becoming more and more efficient. Only a few systems try to discover more complex correspondences or compute different relations other than *equivalence*, such as *subsumption*, *disjointness* or named relations. In recent years, there has been a growing interest in the complex matching because it is one of the future directions of the ontology matching area that aims to support more elaborated and sophisticated approaches. Therefore, the integration of several ontologies with complex correspondences is a big challenge, as neither complex matching nor holistic matching is well addressed in the ontology matching area.

Further, most ontology integration approaches focus on dealing with a particular domain and cannot work with input ontologies that belong to a domain other than the one that has been predefined. This is due to their underlying ontology matching approach that does not apply to diverse domains. Only a few ontology matching tools aim at being generic and therefore suitable for several input domains or any input domain.

Finally, ontologies produced on the Web can be incrementally large. It is necessary to rely on scalable techniques in the ontology integration process. The problem of integrating large ontologies still needs to be better addressed because only a few ontology matching systems are capable of adequately handling violations in such a large matching space, *e.g.*, the *ServOMap* tool [125]. Along the same line, scalability issues are only rarely considered and addressed by state-of-the-art ontology integration works.

9.4. Ontology Integration Evaluation

Despite the endeavor of the OAEI initiative in the creation of ontology matching benchmarks, the ontology integration area is still considerably lacking benchmarks. Ontology integration approaches often perform different integration types, use many parameters as input, make experiments on different input ontologies (two or several ontologies), and apply various evaluation metrics. As a consequence, integration results will considerably differ (depending on the chosen ontologies, integration type, parameters, and metrics). This situation makes a fair comparison between different integration tools even harder. To make future works on ontology integration comparable, it would be compelling to have a common selection of benchmarks. Such benchmarks would serve as a grand basis for evaluating ontology integration approaches in both the qualitative and quantitative performances. The creation of benchmarks can be a thriving factor in the promotion of the ontology integration area. Unlike the ontology integration area, this has been done for a long time in the ontology matching area, notably thanks to the annual OAEI campaign that has been carried out since 2004.

We conclude that the ontology integration problem depends on the performance of the ontology matching module and the quality of its generated alignments, *i.e.*, the quality of the alignment resulting from the ontology matching process will have a direct influence on the quality of the integrated output ontology. We believe that the future of the ontology integration area is closely dependent on the future of the ontology matching area. In addition, it is clear that the creation of benchmarks can drive innovation in both the ontology integration area and the ontology matching area.

10. Conclusion

This survey stands at the crossroads of the domains of ontology matching and ontology integration. We reported consensual definitions of *ontology integration* and *ontology merging*, which are two close and similar notions, and we identified three types of ontology integration, namely the *simple merge*, the *full merge*, and the *asymmetric merge* (or the *ontology enrichment*). This survey has revealed that the unique type of integration that can satisfy all ontology

integration principles is the *ontology enrichment* case. We also described the general principles that should be ideally fulfilled by ontology integration approaches, and reported different strategies to repair the potential issues that can arise in an output integrated ontology. Next, we outlined the most relevant ontology integration research works, starting from the pioneering works to the most recent ones. Moreover, we have shown that there is a large number of research works that integrate different input ontologies and use different evaluation measures to assess the quality of their resulting integrated ontology. In an attempt to standardize the evaluation methods, we gathered all evaluation measures from different works. The investigation on ontology integration benchmarks has just started and might receive increasing attention in the near future. Last but not least, we explored the case of integrating a network of ontologies (*i.e.*, the case of integrating multiple ontologies using pairwise alignments) in order to investigate its particular issues and challenges. We believe that there are promising research directions in repairing *conservativity* violations in general, repairing *coherence* violations in networks of ontologies, creating recognized benchmarks—at least for two small ontologies, for any integration type—and dealing with large ontologies and scalability issues. We hope this brief exploration can provide new insights into future works in the areas of ontology matching and integration.

References

[1] J. De Bruijn, M. Ehrig, C. Feier, F. Martín-Recuerda, F. Scharffe, M. Weiten, Ontology mediation, merging and aligning, Semantic Web Technologies (2006) 95–113. doi:10.1002/047003033X.ch6.

[2] L. Predoiu, C. Feier, F. Scharffe, J. de Bruijn, F. Martín-Recuerda, D. Manov, M. Ehrig, State-of-the-art survey on ontology merging and aligning V2, EU-IST Integrated Project IST-2003-506826 SEKT (2005) 79.

[3] G. Diallo, Efficient building of local repository of distributed ontologies, in: 2011 Seventh International Conference on Signal Image Technology Internet-Based Systems, 2011, pp. 159–166. doi:10.1109/SITIS.2011.45.

[4] H. S. Pinto, A. Gómez-Pérez, J. P. Martins, Some issues on ontology integration, in: the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods, Vol. 18, CEUR Workshop Proceedings, 1999, pp. 7–1–7–12.

[5] H. S. Pinto, Towards Ontology Reuse, in: Proceedings of AAAI99's Workshop on Ontology Management, WS-99, Vol. 13, 1999, pp. 67–73.

[6] H. S. Pinto, J. P. Martins, Ontologies: How can they be built?, Knowledge and information systems 6 (4) (2004) 441–464.

[7] P. Lambrix, H. Tan, SAMBO - A system for aligning and merging biomedical ontologies, Journal on Web Semantics 4 (3) (2006) 196–206. doi:10.1016/j.websem.2006.05.003.

[8] S. K. Kumar, J. A. Harding, Description logic–based knowledge merging for concrete-and fuzzy-domain ontologies, Journal of Engineering Manufacture 230 (5) (2016) 954–971. doi:10.1177/0954405414564404.

[9] K. Dramé, G. Diallo, F. Delva, J. F. Dartigues, E. Mouillet, R. Salamon, F. Mougin, Reuse of termino-ontological resources and text corpora for building a multilingual domain ontology: An application to alzheimer's disease, Journal of Biomedical Informatics 48 (2014) 171 – 182. doi:https://doi.org/10.1016/j.jbi.2013.12.013.

[10] P. A. Bernstein, J. Madhavan, E. Rahm, Generic schema matching, ten years later, PVLDB 4 (11) (2011) 695–701.

[11] Y. Kalfoglou, W. M. Schorlemmer, Ontology Mapping: The state of the art, The knowledge engineering review 18 (1) (2003) 1–31. doi:10.1017/S0269888903000651.

[12] N. Choi, I. Song, H. Han, A survey on ontology mapping, ACM SIGMOD Record 35 (3) (2006) 34–41. doi:10.1145/1168092.1168097.

[13] J. Euzenat, P. Shvaiko, Ontology Matching, Springer, 2007. doi:10.1007/978-3-540-49612-0.

[14] M. Granitzer, V. Sabol, K. W. Onn, D. Lukose, K. Tochtermann, Ontology alignment - A survey with focus on visually supported semi-automatic techniques, Future Internet 2 (3) (2010) 238–258. doi:10.3390/fi2030238.

[15] E. Rahm, Towards large-scale schema and ontology matching, in: Schema Matching and Mapping, Data-Centric Systems and Applications, Springer, 2011, pp. 3–27. doi:10.1007/978-3-642-16518-4_1.

[16] P. Shvaiko, J. Euzenat, Ontology matching: State of the art and future challenges, IEEE Transactions on knowledge and Data Engineering 25 (1) (2013) 158–176. doi:10.1109/TKDE.2011.253.

[17] J. Euzenat, P. Shvaiko, Ontology Matching, Second Ed., Springer, 2013.

[18] L. Otero-Cerdeira, F. J. Rodríguez-Martínez, A. Gómez-Rodríguez, Ontology matching: A literature review, Expert Systems with Applications 42 (2) (2015) 949–971. doi:10.1016/j.eswa.2014.08.032.

[19] P. Ochieng, S. Kyanda, Large-scale ontology matching: State-of-the-art analysis, ACM Computing Surveys 51 (4) (2018) 75:1–75:35. doi:10.1145/3211871.

[20] R. Studer, V. R. Benjamins, D. Fensel, et al., Knowledge Engineering: Principles and Methods, Data and knowledge engineering 25 (1) (1998) 161–198. doi:10.1016/S0169-023X(97)00056-6.

[21] O. Lassila, R. R. Swick, W. Wide, W. Consortium, Resource description framework (RDF) model and syntax specification, citeseer (1998).

[22] E. Miller, An introduction to the resource description framework, Bulletin of the American Society for Information Science and Technology 25 (1) (1998) 15–19. doi:10.1002/bult.105.

[23] D. Brickley, R. V. Guha, A. Layman, Resource description framework (RDF) schema specification, technical report, W3C, 1999. W3C Proposed Recommendation. (1999).

[24] S. Bechhofer, F. Van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, L. A. Stein, et al., OWL web ontology language reference, W3C recommendation (February 2004).
URL https://www.w3.org/TR/owl-ref/

[25] B. Motik, P. F. Patel-Schneider, B. Parsia, C. Bock, A. Fokoue, P. Haase, R. Hoekstra, I. Horrocks, A. Ruttenberg, U. Sattler, et al., OWL 2

web ontology language: Structural specification and functional-style syntax, W3C recommendation (December 2012). URL https://www.w3.org/TR/owl2-syntax/
[26] L. Zhang, J. Ren, X. Li, OIM-SM: A method for ontology integration based on semantic mapping, Journal of Intelligent and Fuzzy Systems 32 (3) (2017) 1983–1995. doi:10.3233/JIFS-161553.
[27] S. Raunich, E. Rahm, Towards a benchmark for ontology merging, in: Proceedings of the OTM Confederated International Workshops: On the Move to Meaningful Internet Systems, Vol. 7567, Springer, 2012, pp. 124–133. doi:10.1007/978-3-642-33618-8_20.
[28] D. L. McGuinness, F. Van Harmelen, et al., OWL web ontology language overview, W3C recommendation (February 2004). URL https://www.w3.org/TR/owl-features/
[29] T. Berners-Lee, J. Hendler, O. Lassila, The semantic web, Scientific American 284 (5) (2001) 34–43.
[30] N. Shadbolt, T. Berners-Lee, W. Hall, The semantic web revisited, IEEE Intelligent Systems 21 (3) (2006) 96–101. doi:10.1109/MIS.2006.62.
[31] F. Baader, D. Calvanese, D. McGuinness, P. Patel-Schneider, D. Nardi, The description logic handbook: Theory, implementation and applications, Cambridge University Press, 2003.
[32] M. Cheatham, C. Pesquita, Semantic Data Integration, in: Handbook of Big Data Technologies, Springer, 2017, pp. 263–305. doi:10.1007/978-3-319-49340-4_8.
[33] É. Thiéblin, O. Haemmerlé, N. Hernandez, C. T. dos Santos, Towards a complex alignment evaluation dataset., in: OM@ ISWC, 2017, pp. 217–218.
[34] É. Thiéblin, O. Haemmerlé, C. Trojahn, Complex matching based on competency questions for alignment: A first sketch, in: Proc. of the International Workshop on Ontology Matching co-located with the International Semantic Web Conference, OM@ISWC, Vol. 2288, 2018, pp. 66–70.
[35] E. Rahm, The case for holistic data integration, in: Proc. of the East European Conference on Advances in Databases and Information Systems, Springer, 2016, pp. 11–27. doi:10.1007/978-3-319-44039-2_2.
[36] I. Megdiche, O. Teste, C. Trojahn, An extensible linear approach for holistic ontology matching, in: the International Semantic Web Conference (ISWC), Springer, 2016, pp. 393–410. doi:10.1007/978-3-319-46523-4_24.
[37] T. Gruetze, C. Böhm, F. Naumann, Holistic and scalable ontology alignment for linked open data., WWW2012 Workshop on Linked Data on the Web (LDOW) 937 (2012).
[38] C. Pesquita, M. Cheatham, D. Faria, J. Barros, E. Santos, F. M. Couto, Building reference alignments for compound matching of multiple ontologies using obo cross-products, in: the 9th International Workshop on Ontology Matching collocated with the 13th ISWC, 2014, pp. 172–173.
[39] D. P. d. S. Oliveira, Compound matching of biomedical ontologies, Ph.D. thesis, Universidade De Lisboa, Faculdade De Ciencias (2015).
[40] J. David, J. Euzenat, F. Scharffe, C. T. dos Santos, The alignment API 4.0, Semantic Web 2 (1) (2011) 3–10. doi:10.3233/SW-2011-0028.
[41] C. Meilicke, H. Stuckenschmidt, A. Tamilin, Reasoning support for mapping revision, Journal of logic and computation 19 (5) (2008) 807–829. doi:10.1093/logcom/exn047.
[42] G. Stoilos, D. Geleta, J. Shamdasani, M. Khodadadi, A novel approach and practical algorithms for ontology integration, in: Proc. of ISWC, Springer, 2018, pp. 458–476. doi:10.1007/978-3-030-00671-6_27.
[43] D. Faria, C. Pesquita, E. Santos, M. Palmonari, I. F. Cruz, F. M. Couto, The AgreementMakerLight ontology matching system, in: Proc. of the OTM 2013 Conferences - Confederated International Conferences, Springer, 2013, pp. 527–541. doi:10.1007/978-3-642-41030-7_38.
[44] A. Solimando, E. Jimenez-Ruiz, G. Guerrini, Minimizing conservativity violations in ontology alignments: Algorithms and evaluation, Knowledge and Information Systems 51 (3) (2017) 775–819. doi:10.1007/s10115-016-0983-3.
[45] A. Zimmermann, C. Le Duc, Reasoning with a network of aligned ontologies, in: the International Conference on Web Reasoning and Rule Systems, Springer, 2008, pp. 43–57. doi:10.1007/978-3-540-88737-9_5.
[46] C. Le Duc, M. Lamolle, A. Zimmermann, An API for distributed reasoning on networked ontologies with alignments., in: Proceedings of the International Conference on Knowledge Engineering and Ontology Development, (KEOD), 2010, pp. 295–304.
[47] L. Serafini, A. Borgida, A. Tamilin, Aspects of distributed and modular ontology reasoning, in: Proc. of IJCAI, Vol. 5, 2005, pp. 570–575.
[48] P. Hitzler, M. Krötzsch, M. Ehrig, Y. Sure, What is ontology merging?, in: American Association for Artificial Intelligence, AAAI Press, 2005, p. 4.
[49] W. Lou, R. Pi, H. Wang, Y. Ju, Low-cost similarity calculation on ontology fusion in knowledge bases, Journal of Information Science. (2019) 14doi:10.1177/0165551519870456.
[50] P. Mitra, G. Wiederhold, M. Kersten, A graph-oriented model for articulation of ontology interdependencies, in: Proc. of the International Conference on Extending Database Technology, Springer, 2000, pp. 86–100. doi:10.1007/3-540-46439-5_6.
[51] C. Meilicke, H. Stuckenschmidt, An efficient method for computing alignment diagnoses, in: International Conference on Web Reasoning and Rule Systems, 2009, pp. 182–196. doi:10.1007/978-3-642-05082-4_13.
[52] J. Kim, M. Jang, Y.-G. Ha, J.-C. Sohn, S. J. Lee, MoA: OWL ontology merging and alignment tool for the semantic web, in: the International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, 2005, pp. 722–731. doi:10.1007/11504894_100.
[53] J. Heflin, J. Hendler, Dynamic Ontologies on the Web, in: Proc. of AAAI/IAAI, AAAI Press / The MIT Press, 2000, pp. 443–449.
[54] M. Uschold, Achieving semantic interoperability using RDF and OWL-v10, 2005, Last accessed 13 Nov 2020 (2005). URL https://www.w3.org/2001/sw/BestPractices/OEP/SemInt/
[55] E. Jiménez-Ruiz, B. C. Grau, I. Horrocks, R. B. Llavori, Ontology Integration Using Mappings: Towards Getting the Right Logical Consequences, in: Proc. of the 6th European Semantic Web Conference (ESWC), Springer, 2009, pp. 173–187. doi:10.1007/978-3-642-02121-3_16.
[56] P. Bouquet, F. Giunchiglia, F. Van Harmelen, L. Serafini, H. Stuckenschmidt, C-OWL: Contextualizing ontologies, in: Proc. of the International Semantic Web Conference (ISWC), Springer, 2003, pp. 164–179. doi:10.1007/978-3-540-39718-2_11.
[57] A. Borgida, L. Serafini, Distributed description logics: Assimilating information from peer sources, Journal on data semantics I 1 (2003) 153–184. doi:10.1007/978-3-540-39733-5_7.

[58] B. C. Grau, B. Parsia, E. Sirin, Working with multiple ontologies on the semantic web, in: the International Semantic Web Conference (ISWC), Springer, 2004, pp. 620–634. doi:10.1007/978-3-540-30475-3_43.
[59] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, et al., SWRL: A semantic web rule language combining OWL and RuleML, W3C Member submission 21 (79) (2004) 1–31.
[60] A. Miles, S. Bechhofer, SKOS simple knowledge organization system reference, W3C recommendation 18 (2009) W3C.
[61] C. Meilicke, Alignment Incoherence in Ontology Matching, Ph.D. thesis, Universität Mannheim (2011).
[62] J. Sowa, Electronic communication in the onto-std mailing list (December 1997).
[63] S. Raunich, E. Rahm, Atom: Automatic target-driven ontology merging, in: Proc. of the 27th International Conference on Data Engineering, ICDE 2011, IEEE, 2011, pp. 1276–1279. doi:10.1109/ICDE.2011.5767871.
[64] S. Raunich, E. Rahm, Target-driven Merging of Taxonomies with ATOM, Information Systems 42 (2014) 1–14. doi:10.1016/j.is.2013.11.001.
[65] R. Pottinger, P. A. Bernstein, Merging models based on given correspondences, in: Proceedings of the 29th International Conference on Very Large Data Bases (VLDB 2003), Morgan Kaufmann, 2003, pp. 826–873. doi:10.1016/B978-012722442-8/50081-1.
[66] S. Babalou, B. König-Ries, GMRs: Reconciliation of generic merge requirements in ontology integration, in: Proc. of the Posters&Demos Track of the International Conference on Semantic Systems (SEMANTiCS), Vol. 2451, 2019, p. 5.
URL http://ceur-ws.org/Vol-2451/paper-04.pdf
[67] U. Sattler, R. Stevens, P. Lord, (I can't get no) satisfiability, Ontogenesis, Last accessed 13 Nov 2020 (2013).
URL http://ontogenesis.knowledgeblog.org/1329
[68] S. Bail, Common reasons for ontology inconsistency, Ontogenesis, Last accessed 13 Nov 2020 (2013).
URL http://ontogenesis.knowledgeblog.org/1343
[69] M. Fahad, N. Moalla, A. Bouras, M. A. Qadir, M. Farukh, Disjoint-Knowledge Analysis and Preservation in Ontology Merging Process, in: Proc. of the Fifth International Conference on Software Engineering Advances, ICSEA, IEEE, 2010, pp. 422–428. doi:10.1109/ICSEA.2010.72.
[70] A. Solimando, E. Jiménez-Ruiz, G. Guerrini, Detecting and correcting conservativity principle violations in ontology-to-ontology mappings, in: Proc. of the International Semantic Web Conference (ISWC), Springer, 2014, pp. 1–16. doi:10.1007/978-3-319-11915-1_1.
[71] A. Solimando, E. Jiménez-Ruiz, G. Guerrini, A multi-strategy approach for detecting and correcting conservativity principle violations in ontology alignments, in: the 11th International Workshop on OWL: Experiences and Directions (OWLED) co-located with ISWC, 2014, pp. 13–24.
[72] M. Poveda-Villalón, M. C. Suárez-Figueroa, A. Gómez-Pérez, Validating ontologies with OOPS!, in: Proceedings of the 18th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2012), Vol. 7603, Springer, 2012, pp. 267–281. doi:10.1007/978-3-642-33876-2_24.
[73] M. Poveda-Villalón, A. Gómez-Pérez, M. C. Suárez-Figueroa, OOPS! (OntOlogy Pitfall Scanner!): An on-line tool for ontology evaluation, International Journal on Semantic Web and Information Systems (IJSWIS) 10 (2) (2014) 7–34. doi:10.4018/ijswis.2014040102.
[74] F. Duchateau, Z. Bellahsene, Measuring the quality of an integrated schema, in: Proceedings of the International Conference on Conceptual Modeling, Springer, 2010, pp. 261–273. doi:10.1007/978-3-642-16373-9_19.
[75] N. F. Noy, M. A. Musen, PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment, in: Proc. of AAAI/IAAI, AAAI Press / The MIT Press, 2000, pp. 450–455.
[76] A. Guzmán-Arenas, A. Cuevas-Rasgado, Knowledge accumulation through automatic merging of ontologies, Expert Systems with Applications 37 (3) (2010) 1991–2005. doi:10.1016/j.eswa.2009.06.078.
[77] J. Zhang, Y. Lv, An approach of refining the merged ontology, in: Proceedings of the 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2012), IEEE, 2012, pp. 802–806. doi:10.1109/FSKD.2012.6233973.
[78] S. Babalou, B. König-Ries, Towards building knowledge by merging multiple ontologies with *CoMerger*: A partitioning-based approach, CoRR abs/2005.02659 (2020). arXiv:2005.02659.
[79] A. L. Rector, N. Drummond, M. Horridge, J. Rogers, H. Knublauch, R. Stevens, H. Wang, C. Wroe, OWL pizzas: Practical experience of teaching OWL-DL: common errors & common patterns, in: Proceedings of the 14th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2004), Vol. 3257, Springer, 2004, pp. 63–81. doi:10.1007/978-3-540-30202-5_5.
[80] G. Schreiber, OWL restrictions, Last accessed 13 Nov 2020 (May 2005).
URL https://www.cs.vu.nl/~guus/public/owl-restrictions/
[81] M. Horridge, H. Knublauch, A. Rector, R. Stevens, C. Wroe, A practical guide to building OWL ontologies using the protégé-OWL plugin and CO-ODE tools edition 1.0, University of Manchester (2004).
URL https://cse.buffalo.edu/~shapiro/Courses/CSE663/Fall07/ProtegeOWLTutorial.pdf
[82] C. Meilicke, H. Stuckenschmidt, Applying logical constraints to ontology matching, in: the Annual Conference on Artificial Intelligence, Springer, 2007, pp. 99–113. doi:10.1007/978-3-540-74565-5_10.
[83] E. Jiménez-Ruiz, B. C. Grau, LogMap: Logic-based and scalable ontology matching, in: the 10th International Semantic Web Conference (ISWC), Vol. 7031, 2011, pp. 273–288. doi:10.1007/978-3-642-25073-6_18.
[84] E. Jiménez-Ruiz, B. C. Grau, Y. Zhou, I. Horrocks, Large-scale interactive ontology matching: Algorithms and implementation, in: the 20th European Conference on Artificial Intelligence ECAI, Vol. 242, IOS Press, 2012, pp. 444–449. doi:10.3233/978-1-61499-098-7-444.
[85] E. Santos, D. Faria, C. Pesquita, F. M. Couto, Ontology alignment repair through modularization and confidence-based heuristics, PloS one 10 (12) (2015) e0144807. doi:10.1371/journal.pone.0144807.
[86] D. Ngo, Z. Bellahsene, YAM++: A multi-strategy based approach for ontology matching task, in: Proceedings of the International Conference on Knowledge Engineering and Knowledge Management, Springer, 2012, pp. 421–425. doi:10.1007/978-3-642-33876-2_38.
[87] Y. R. Jean-Mary, E. P. Shironoshita, M. R. Kabuka, Ontology matching with semantic verification, Journal of Web Semantics 7 (3) (2009) 235–251. doi:10.1016/j.websem.2009.04.001.
[88] P. Wang, Y. Zhou, B. Xu, Matching large ontologies based on reduction anchors, in: Proc. of the Twenty-Second International Joint Conference on Artificial Intelligence, (IJCAI), 2011, pp. 2343–2348. doi:10.5591/978-1-57735-516-8/IJCAI11-390.

[89] S. Schlobach, R. Cornet, et al., Non-standard reasoning services for the debugging of description logic terminologies, in: Proc. of IJCAI, Vol. 3, 2003, pp. 355–362.
[90] F. Baader, S. Brandt, C. Lutz, Pushing the EL envelope, in: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence IJCAI-05, Professional Book Center, 2005, pp. 364–369.
[91] F. Baader, C. Lutz, S. Brandt, Pushing the EL envelope further, in: Proceedings of the Fourth OWLED Workshop on OWL: Experiences and Directions, Vol. 496 of CEUR Workshop Proceedings, 2008, pp. 1–10.
[92] B. Konev, D. Walther, F. Wolter, The logical difference problem for description logic terminologies, in: Proc. of the International Joint Conference on Automated Reasoning (IJCAR), Springer, 2008, pp. 259–274. doi:10.1007/978-3-540-71070-7_21.
[93] R. Kontchakov, F. Wolter, M. Zakharyaschev, Can you tell the difference between DL-Lite ontologies?, in: Proceedings of the Eleventh International Conference on the Principles of Knowledge Representation and Reasoning KR, AAAI Press, 2008, pp. 285–295.
[94] D. Gale, L. S. Shapley, College admissions and the stability of marriage, The American Mathematical Monthly 69 (1) (1962) 9–15. doi:10.1080/00029890.1962.11989827.
[95] A. Tordai, On combining alignment techniques, Ph.D. thesis, Vrije Universiteit, Amsterdam, The Netherlands (2012).
[96] P. Arnold, E. Rahm, Semantic enrichment of ontology mappings: A linguistic-based approach, in: Proc. of the East European Conference on Advances in Databases and Information Systems, Springer, 2013, pp. 42–55. doi:10.1007/978-3-642-40683-6_4.
[97] P. Lambrix, Z. Dragisic, V. Ivanova, Get my pizza right: Repairing missing is-a relations in $\mathcal{ALC}$ ontologies, in: Proceedings of the Second Joint International Semantic Technology Conference (JIST), Springer, 2012, pp. 17–32. doi:10.1007/978-3-642-37996-3_2.
[98] P. Lambrix, F. Wei-Kleiner, Z. Dragisic, V. Ivanova, Repairing missing is-a structure in ontologies is an abductive reasoning problem, in: Proc. of the Second International Workshop on Debugging Ontologies and Ontology Mappings (WoDOOM 2013), 2013, pp. 33–44.
[99] S. Babalou, B. König-Ries, Towards multiple ontology merging with $\mathcal{C}o\mathcal{M}$erger, in: Proceedings of the 19th International Semantic Web Conference (ISWC 2020) - Demos and Industry Tracks, Vol. 2721, CEUR-WS.org, 2020, pp. 59–64.
URL http://ceur-ws.org/Vol-2721/paper500.pdf
[100] S. Babalou, A. Algergawy, B. Lantow, B. König-Ries, Why the mapping process in ontology integration deserves attention, in: the International Conference on Information Integration and Web-based Applications & Services (iiWAS), ACM, 2017, pp. 451–456. doi:10.1145/3151759.3151834.
[101] M. Kachroudi, G. Diallo, S. Ben Yahia, OAEI 2017 Results of KEPLER, in: Proceedings of the 12th International Workshop on Ontology Matching (OM) co-located with the 16th International Semantic Web Conference (ISWC), CEUR-WS.org, 2017, pp. 138–145.
[102] P. Ziemba, J. Jankowski, J. Watrobski, W. Wolski, J. Becker, Integration of Domain Ontologies in the Repository of Website Evaluation Methods, in: Proc. of the Federated Conference on Computer Science and Information Systems (FedCSIS), IEEE, 2015, pp. 1585–1595. doi:10.15439/2015F297.
[103] P. Mitra, G. Wiederhold, S. Decker, A Scalable Framework for the Interoperation of Information Sources, in: Proceedings of SWWS'01, The first Semantic Web Working Symposium, 2001, pp. 317–329.
[104] C. Batini, M. Lenzerini, S. B. Navathe, A comparative analysis of methodologies for database schema integration, ACM Computing Surveys (CSUR) 18 (4) (1986) 323–364. doi:10.1145/27633.27634.
[105] J. Euzenat, Revision in networks of ontologies, Artificial intelligence 228 (2015) 195–216. doi:10.1016/j.artint.2015.07.007.
[106] N. Chatterjee, N. Kaushik, D. Gupta, R. Bhatia, Ontology Merging: A Practical Perspective, in: Proc. of the International Conference on Information and Communication Technology for Intelligent Systems, Springer, 2017, pp. 136–145. doi:10.1007/978-3-319-63645-0_15.
[107] E. G. Caldarola, A. M. Rinaldi, An approach to ontology integration for ontology reuse, in: the International Conference on Information Reuse and Integration, IRI, IEEE, 2016, pp. 384–393. doi:10.1109/IRI.2016.58.
[108] N. F. Noy, M. A. Musen, The PROMPT suite: interactive tools for ontology merging and mapping, International Journal of Human-Computer Studies 59 (6) (2003) 983–1024. doi:10.1016/j.ijhcs.2003.08.002.
[109] D. L. McGuinness, R. Fikes, J. Rice, S. Wilder, The Chimaera Ontology Environment, in: Proc. of AAAI/IAAI, Vol. 2000, AAAI Press / The MIT Press, 2000, pp. 1123–1124.
[110] G. Stumme, A. Maedche, FCA-MERGE: Bottom-Up merging of ontologies, in: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI), Morgan Kaufmann, 2001, pp. 225–234.
[111] D. Dou, D. V. McDermott, P. Qi, Ontology Translation on the Semantic Web, in: Proc. of the OTM Confederated International Conferences, CoopIS, DOA, and ODBASE: On the Move to Meaningful Internet Systems, Springer, 2003, pp. 952–969. doi:10.1007/978-3-540-39964-3_60.
[112] K. Kotis, G. A. Vouros, K. Stergiou, Towards automatic merging of domain ontologies: The HCONE-merge approach, Journal on Web semantics 4 (1) (2006) 60–79. doi:10.1016/j.websem.2005.09.004.
[113] O. Udrea, L. Getoor, R. J. Miller, Leveraging Data and Structure in Ontology Integration, in: the SIGMOD International Conference on Management of Data, ACM, 2007, pp. 449–460. doi:10.1145/1247480.1247531.
[114] M. Maree, M. Belkhatir, A coupled statistical/semantic framework for merging heterogeneous domain-specific ontologies, in: Proc. of the 22nd IEEE International Conference on Tools with Artificial Intelligence, (ICTAI 2010), Vol. 2, IEEE, 2010, pp. 159–166. doi:10.1109/ICTAI.2010.138.
[115] M. Maree, M. Belkhatir, Addressing semantic heterogeneity through multiple knowledge base assisted merging of domain-specific ontologies, Knowledge-Based Systems 73 (2015) 199–211. doi:10.1016/j.knosys.2014.10.001.
[116] M. Fahad, N. Moalla, A. Bouras, Detection and resolution of semantic inconsistency and redundancy in an automatic ontology merging system, Journal of Intelligent Information Systems. 39 (2) (2012) 535–557. doi:10.1007/s10844-012-0202-y.
[117] L. Zhao, R. Ichise, Ontology Integration for Linked Data, Journal on Data Semantics 3 (4) (2014) 237–254. doi:10.1007/s13740-014-0041-9.
[118] Z. El Jerroudi, J. Ziegler, iMERGE: Interactive ontology merging, in: Proceedings of the International Conference on Knowledge Engineering and Knowledge Management (EKAW 2008), Springer, 2008, pp. 52–56.
[119] M. Mahfoudh, G. Forestier, M. Hassenforder, A Benchmark for Ontologies Merging Assessment, in: Proc. of the International Conference on

Knowledge Science, Engineering and Management (KSEM), Springer, 2016, pp. 555–566. doi:10.1007/978-3-319-47650-6_44.
[120] M. Horridge, S. Bechhofer, The OWL API: A Java API for OWL ontologies, Semantic Web 2 (1) (2011) 11–21. doi:10.3233/SW-2011-0025.
[121] O. Bodenreider, The Unified Medical Language System (UMLS): Integrating biomedical terminology, Nucleic acids research 32 (Database-Issue) (2004) 267–270. doi:10.1093/nar/gkh061.
[122] Y. Kazakov, M. Krötzsch, F. Simancik, ELK Reasoner: Architecture and Evaluation., in: Proceedings of the 1st International Workshop on OWL Reasoner Evaluation (ORE), Vol. 858, CEUR-WS.org, 2012, p. 12.
[123] C. Pesquita, D. Faria, E. Santos, F. M. Couto, To repair or not to repair: Reconciling correctness and coherence in ontology reference alignments, in: the Ontology Matching Workshop (OM) co-located with the International Semantic Web Conference (ISWC), CEUR-WS.org, 2013, pp. 13–24.
[124] E. Jiménez-Ruiz, B. C. Grau, I. Horrocks, R. Berlanga, Logic-based assessment of the compatibility of UMLS ontology sources, Journal of Biomedical Semantics 2 (1) (2011) S2.
[125] G. Diallo, An effective method of large scale ontology matching, Journal of Biomedical Semantics 5 (1) (2014) 44. doi:10.1186/2041-1480-5-44.